

BBBBBBBBBBBBBBB AAAAAAAA SSSSSSSSSSSS RRRRRRRRRRRRR TTTTTTTTTTTTTT LLL
BBBBBBBBBBBBBBB AAAAAAAA SSSSSSSSSSSS RRRRRRRRRRRRR TTTTTTTTTTTTTT LLL
BBBBBBBBBBBBBBB AAAAAAAA SSSSSSSSSSSS RRRRRRRRRRRRR TTTTTTTTTTTTTT LLL
BBB BBB AAA AAA SSS RRR RRR TTT LLL
BBB BBB AAA AAA SSS RRR RRR TTT LLL
BBB BBB AAA AAA SSS RRR RRR TTT LLL
BBB BBB AAA AAA SSS RRR RRR TTT LLL
BBB BBB AAA AAA SSS RRR RRR TTT LLL
BBB BBB AAA AAA SSS RRR RRR TTT LLL
BBB BBB AAA AAA SSS RRR RRR TTT LLL
BBBBBBBBBBBBBBB AAA AAA SSSSSSSSSS RRRRRRRRRRRRR TTT LLL
BBBBBBBBBBBBBBB AAA AAA SSSSSSSSSS RRRRRRRRRRRRR TTT LLL
BBBBBBBBBBBBBBB AAA AAA SSSSSSSSSS RRRRRRRRRRRRR TTT LLL
BBB BBB AAAAAAAAAAAAAA SSS RRR RRR TTT LLL
BBB BBB AAAAAAAAAAAAAA SSS RRR RRR TTT LLL
BBB BBB AAAAAAAAAAAAAA SSS RRR RRR TTT LLL
BBB BBB AAA AAA SSS RRR RRR TTT LLL
BBB BBB AAA AAA SSS RRR RRR TTT LLL
BBB BBB AAA AAA SSS RRR RRR TTT LLL
BBBBBBBBBBBBBBB AAA AAA SSSSSSSSSSSS RRR RRR TTT LLLL
BBBBBBBBBBBBBBB AAA AAA SSSSSSSSSSSS RRR RRR TTT LLLL
BBBBBBBBBBBBBBB AAA AAA SSSSSSSSSSSS RRR RRR TTT LLLL

| | | | | | | | | |
|------------|---------|----------|----------|-----------|----------|----------|----------|--------|
| BBBBBBBBBB | AAAAAA | SSSSSSSS | RRRRRRRR | EEEEEEEEE | CCCCCCCC | PPPPPPPP | RRRRRRRR | 000000 |
| BBBBBBBBBB | AAAAAA | SSSSSSSS | RRRRRRRR | EEEEEEEEE | CCCCCCCC | PPPPPPPP | RRRRRRRR | 000000 |
| BB BB | AA AA | SS | RR RR | EE | CC | PP PP | RR RR | 00 00 |
| BB BB | AA AA | SS | RR RR | EE | CC | PP PP | RR RR | 00 00 |
| BB BB | AA AA | SS | RR RR | EE | CC | PP PP | RR RR | 00 00 |
| BB BB | AA AA | SS | RR RR | EE | CC | PP PP | RR RR | 00 00 |
| BBBBBBBBBB | AA AA | SSSSSS | RRRRRRRR | EEEEEEEEE | CC | PPPPPPPP | RRRRRRRR | 00 00 |
| BBBBBBBBBB | AA AA | SSSSSS | RRRRRRRR | EEEEEEEEE | CC | PPPPPPPP | RRRRRRRR | 00 00 |
| BB BB | AAAAAAA | SS | RR RR | EE | CC | PP | RR RR | 00 00 |
| BB BB | AAAAAAA | SS | RR RR | EE | CC | PP | RR RR | 00 00 |
| BB BB | AA AA | SS | RR RR | EE | CC | PP | RR RR | 00 00 |
| BB BB | AA AA | SS | RR RR | EE | CC | PP | RR RR | 00 00 |
| BBBBBBBBBB | AA AA | SSSSSSSS | RR RR | EEEEEEEEE | CCCCCCCC | PP | RR RR | 000000 |
| BBBBBBBBBB | AA AA | SSSSSSSS | RR RR | EEEEEEEEE | CCCCCCCC | PP | RR RR | 000000 |

| | | |
|------------|--|----------|
| LL | | SSSSSSSS |
| LL | | SSSSSSSS |
| LL | | SS |
| LLLLLLLLLL | | SSSSSSSS |
| LLLLLLLLLL | | SSSSSSSS |

```
1 0001 0 MODULE BASS$REC_PROC (          ! Record processing level of abstraction
2 0002 0           IDENT = '1-095'      ! File: BASRECPRO.B32 Edit:MDL1095
3 0003 0           ) =
4 0004 1 BEGIN
5
6 0006 1 ****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 ****
28 0028 1 ++
29 0029 1 FACILITY: BASIC Support Library - not user callable
30 0030 1
31 0031 1 ABSTRACT:
32 0032 1
33 0033 1
34 0034 1 This module implements the record processing level of
35 0035 1 abstraction which is the 3rd level and is called only from
36 0036 1 the user data formatter level (2nd level) when the user
37 0037 1 portion of a record buffer is full (WRITE) or empty
38 0038 1 (READ). This module adds any per record formatting (as
39 0039 1 distinguished from per I/O statement or per I/O list element
40 0040 1 formatting) and then calls RMS ($PUT or $GET). RMS errors
41 0041 1 are converted to BASIC errors and are signaled.
42 0042 1
43 0043 1 ENVIRONMENT: User access mode; AST level or not.
44 0044 1
45 0045 1 AUTHOR: Donald G. Petersen; CREATION DATE: 16-Mar-78
46 0046 1
47 0047 1 MODIFIED BY:
48 0048 1
49 0049 1 0-61 - Add ATTEMPT TO READ NON-EXISTANT RECORD error to seq. reads.
50 0050 1 JMT 02-Jan-78
51 0051 1 Donald G. Petersen, 16-Mar-78 : VERSION 1-01
52 0052 1 1-01 - original DGP
53 0053 1 1-02 - Change to JSB linkages. DGF 14-Nov-78
54 0054 1 1-004 - Update copyright notice and add device names to REQUIRE
55 0055 1 files. JBS 29-NOV-78
56 0056 1 1-005 - Add BASS$RECOUNT to support the Basic RECOUNT function. DGP
57 0057 1 03-Dec-78
```

58 0058 1 | 1-006 - Dot problem in BASSRECOUNT. DGP 04-Dec-78
59 0059 1 | 1-007 - Add fudge factor to RECOUNT for the line terminator if input is
60 0060 1 | from a terminal. DGP 04-Dec-78
61 0061 1 | 1-008 - Change REQUIRE file names from FOR... to OTS... JBS 07-DEC-78
62 0062 1 | 1-009 - Call BAS\$SIG\$IGNAL_I0 for RMS errors. JBS 18-DEC-78
63 0063 1 | 1-010 - Add new routines for READ. DGP 19-Dec-78
64 0064 1 | 1-011 - Change references to ISBSA_BUF_PTR, BUF_BEG, BUF_END to LUB.
65 0065 1 | DGP 05-Jan-78
66 0066 1 | 1-012 - Change to CR format for terminal data files. DGP 11-Jan-79
67 0067 1 | 1-013 - Make a few changes for recursive I/O. DGP 15-Jan-79
68 0068 1 | 1-014 - Remove signalling for ^Z. Moved to UDF level for INPUT LINE hand-
69 0069 1 | ling. DGP 15-Jan-79
70 0070 1 | 1-015 - Change stack frame prefix to BSFS. JBS 08-FEB-1979
71 0071 1 | 1-016 - Add BAS\$REC_GSE (Basic GET sequential). DGP 19-Feb-79
72 0072 1 | 1-017 - Add BAS\$REC_PSE (Basic PUT sequential). DGP 20-Feb-79
73 0073 1 | 1-018 - set RAB\$L_RBF in BAS\$REC_PSE. DGP 20-Feb-79
74 0074 1 | 1-019 - Set RAB\$L_RBF in BAS\$GSE. DGP 21-Feb-79
75 0075 1 | 1-020 - Null fill buffer for GET. DGP 27-Feb-79
76 0076 1 | 1-021 - Add REC routines for FIND, DELETE, UPDATE, RESTORE, SCRATCH. DGP
77 0077 1 | 27-Feb-79
78 0078 1 | 1-022 - Add BASIOERR.REQ for error handling. DGP 28-Feb-79
79 0079 1 | 1-023 - Add BASUNLOCK and BASFREE. DGP 28-Feb-79
80 0080 1 | 1-024 - Set RAB\$L_RBF in BAS\$REC_UPD. DGP 01-Mar-79
81 0081 1 | 1-025 - Add REC_PRE, REC_GRE, REC_FRE. DGP 02-Mar-79
82 0082 1 | 1-026 - More work on relative I/O. DGP 05-Mar-79
83 0083 1 | 1-027 - Update pointer for READ in Basic Major Frame in RMF9. DGP 06-Mar-79
84 0084 1 | 1-028 - Add support for Basic "foreign buffers". DGP 27-Mar-79
85 0085 1 | 1-029 - Point all GETs and PUTs off to GET_ERROR or PUT_ERROR. DGP 02-Apr-79
86 0086 1 | 1-030 - Add more routines to support ISAM. DGP 03-Apr-79
87 0087 1 | 1-031 - Fix PUT sequential to support ISAM. DGP 04-Apr-79
88 0088 1 | 1-032 - Put in indexed I/O stuff. 06-Apr-79
89 0089 1 | 1-033 - Bug fixes in indexed. 10-Apr-79 DGP
90 0090 1 | 1-034 - Implement the WAIT statement, using LUB\$L_WAIT_TIME.
91 0091 1 | JBS 10-APR-1979
92 0092 1 | 1-035 - Implement the ECHO and NOECHO functions, using LUB\$V_NOECHO.
93 0093 1 | JBS 17-APR-1979
94 0094 1 | 1-036 - Add code to handle single-character input from GET
95 0095 1 | SEQUENTIAL. JBS 17-APR-1979
96 0096 1 | 1-037 - Implement the CTRLO and RCTRLO functions, using LUB\$V_CCO.
97 0097 1 | JBS 19-APR-1979
98 0098 1 | 1-038 - Implement the Cancel Typeahead function, using LUB\$V_PTA.
99 0099 1 | JBS 01-MAY-1979
100 0100 1 | 1-039 - Add Basic PRINT USING support. DGP 15-May-79
101 0101 1 | 1-040 - Change BIND to GLOBAL BIND ROUTINE in PRINT USING support.
102 0102 1 | JBS 16-MAY-1979
103 0103 1 | 1-041 - Add BAS\$RECOU_INIT. JBS 04-JUN-1979
104 0104 1 | 1-042 - Add REC level for MAT INPUT. DGP 05-Jun-79
105 0105 1 | 1-043 - Clean up a lot and put real code into Matrix Input routines. DGP
106 0106 1 | 14-Jun-79
107 0107 1 | 1-044 - Make REC_MIN1 look for continuation character. DGP 20-Jun-79
108 0108 1 | 1-045 - Terminal devices use PRN format for output. DGP 10-Jul-79
109 0109 1 | 1-046 - Add BAS\$NUM INIT, BAS\$NUM2 INIT, BAS\$MAT_LINPUT, BAS\$MAT_READ,
110 0110 1 | BAS\$NUM, BAS\$NUM. DGP 13-Jul-79
111 0111 1 | 1-047 - Change ISBSL MAJ_F PTR to ISBSA MAJ_F PTR, JBS 24-JUL-1979
112 0112 1 | 1-048 - Signal if READ with no DATA. DGP 07-Aug-79
113 0113 1 | 1-049 - Debug MAT I/O. DGP 07-Aug-79
114 0114 1 | 1-050 - STOP a few errors that are being SIGNALled. DGP 05-Sep-79

115 0115 1 | 1-051 - FREE and UNLOCK are noops if no record locked. DGP 06-Sep-79
116 0116 1 | 1-052 - Move NUM INIT and NUM2 INIT to BASSMAT IO, and move BASS\$BLNK_LINE
117 0117 1 | from BASSMAT IO to here. DGP 06-Sep-79
118 0118 1 | 1-053 - Load LUBSA_RBUF_ADR for GET and PUT for Locate mode (RMS). DGP
119 0119 1 | 13-Sep-79
120 0120 1 | 1-054 - Clear the prompt buffer in GET_ERROR. DGP 17-Sep-79
121 0121 1 | 1-055 - Fix BASS\$BLNK_LINE. DGP 04-Oct-79
122 0122 1 | 1-056 - Add MAT READ. DGP 11-Oct-79
123 0123 1 | 1-057 - Add a REC9 routine for MAT PRINT. DGP 12-Oct-79
124 0124 1 | 1-058 - Add BASS\$REC_MLI1. DGP 12-Oct-79
125 0125 1 | 1-059 - Fix BASS\$REC_WSL1 to leave the cursor alone. DGP 02-Nov-79
126 0126 1 | 1-060 - Allow BASS\$REC_WSL1 to accept an argument. DGP 06-Nov-79
127 0127 1 | 1-061 - GET will only null fill the buffer, if necessary, after a GET. DGP
128 0128 1 | 12-Nov-79
129 0129 1 | 1-062 - BASS\$REC_MPR1 needs an LF if no format char. DGP 13-Nov-79
130 0130 1 | 1-063 - Use LUBSA UBF to simplify foreign buffer code. JBS 13-NOV-1979
131 0131 1 | 1-064 - BASS\$REC_MIN1 should not differentiate between terminal & non-
132 0132 1 | terminal devices. DGP 14-Nov-79
133 0133 1 | 1-065 - GET relative not null filling the buffer properly. DGP 29-Nov-79
134 0134 1 | 1-066 - Null fill the buffer before restoring foreign buffer pointers for
135 0135 1 | GET. DGP 18-Dec-79
136 0136 1 | 1-067 - RMS does not return a terminator in the STV field for files. DGP
137 0137 1 | 03-Jan-80
138 0138 1 | 1-068 - REC_WSL9 should only write a record if the output buffer has some-
139 0139 1 | thing in it to write. DGP 03-Jan-80
140 0140 1 | 1-069 - Addition to 1-068. Should also write a record if there was no element
141 0141 1 | transmitter. DGP 04-Jan-80
142 0142 1 | 1-070 - Unconditionally write a CR in WSL1. DGP 14-Jan-80
143 0143 1 | 1-071 - Restore "foreign buffers" properly and set RECOUNT in GET Indexed
144 0144 1 | and Relative. DGP 12-Feb-80
145 0145 1 | 1-072 - Adjust the Global RECOUNT to include the length of an escape sequence.
146 0146 1 | DGP 22-Feb-80
147 0147 1 | 1-073 - A previous edit to fix a problem with foreign buffers reintroduced
148 0148 1 | a problem with only null padding the buffer after a successful GET.
149 0149 1 | DGP 26-Feb-80
150 0150 1 | 1-074 - REC_WSL9 should set VFC2 to BASS\$NULL (no carriage control) if there
151 0151 1 | is a format character. DGP 26-Feb-80
152 0152 1 | 1-075 - Update the cursor position for INPUT if terminated by an escape.
153 0153 1 | DGP 27-Feb-80
154 0154 1 | 1-076 - When calculating CPOS (current cursor position) following an INPUT
155 0155 1 | statement, take the prompt string into account.
156 0156 1 | 1-077 - REC_RSL1 is not updating the cursor position correctly. DGP 04-Mar-80
157 0157 1 | 1-078 - REC_WSL9 should set the 'pre' carriage control for the next record
158 0158 1 | to LF if there is no format character 'cuz of recursive I/O. DGP
159 0159 1 | 07-Mar-80
160 0160 1 | 1-079 - Rationalize the CCO and PTA bits. CCO is now copied from LUB to RAB
161 0161 1 | when initializing for output, and PTA when initializing for input.
162 0162 1 | JBS 31-MAR-1980
163 0163 1 | 1-080 - Clear the dirty bit [CB [LUB\$V OUTBUF DR] in PUT_ERROR so BASSCLOSE
164 0164 1 | when invoked by the unwind won't get confused and do a PUT.
165 0165 1 | FM 11-SEP-80
166 0166 1 | 1-081 - Tack on the terminator(s) to the buffer when a GET is done on a
167 0167 1 | terminal device file in BASS\$REC_GSE.
168 0168 1 | 1-082 - Add/transfer BASS\$WAIT to this module, now wait routines are part of
169 0169 1 | the sharable image. The routines added are BASS\$WAIT,
170 0170 1 | BASS\$READ_WAIT. We had to make WAIT routines part of the sharable
171 0171 1 | image because WAIT was requested to become a GLOBAL, and routines in

172 0172 1 | this module had to read it.
173 0173 1 | 1-083- Only if LUB\$B_RAT indicates CR format tack on the CRLF. FM 9-feb-81
174 0174 1 | 1-084- Cursor position not updated correctly if INPUT was
175 0175 1 | terminated by an escape - code should check if previous
176 0176 1 | PRINT was terminated by a semicolon or comma. PLL 5-7-81
177 0177 1 | 1-085- The purge typeahead function is no longer setting the PTA bit in
178 0178 1 | the LUB, so BASS\$REC_RSL0, BASS\$REC_MIN0, and BASS\$REC_GSE don't need
179 0179 1 | to check it anymore. PLL 6-Aug-81
180 0180 1 | 1-086 - Add support for RFA access and manual record locking. PLL 1-Jun-82
181 0181 1 | 1-087 - BASS\$REC_GIN and BASS\$REC_FIN should check for a decimal key
182 0182 1 | when setting the key size in the RAB. PLL 6-Jul-1982
183 0183 1 | 1-088 - Add support for ANSI INPUT. If not enough data is supplied, the
184 0184 1 | entire INPUT must be restarted. PLL 29-Jul-1982
185 0185 1 | 1-089 - Fix CTRL/O rationalization. Unconditionally copy whatever its
186 0186 1 | state is in the LUB into the RAB, and clear it in the LUB.
187 0187 1 | MDL and JBS 10-Aug-1982
188 0188 1 | 1-090 - Set buffer pointer (in WSL9) from buffer beginning pointer rather
189 0189 1 | than from RBUF ADR. This fixes the problem of pointing to the
190 0190 1 | wrong buffer when the user enters a CTRL/C while a line is being
191 0191 1 | written, and his control-c routine writes a line also.
192 0192 1 | MDL and PLL 19-Aug-1982
193 0193 1 | 1-091 - In BASS\$REC_RSL1, BASS\$SIGNAL_IO the too little data error instead
194 0194 1 | of calling BASS\$STOP IO. PLL 27-Sep-1982
195 0195 1 | 1-092 - In BASS\$REC_RSL0, the contents of the print buffer should be \$PUT
196 0196 1 | before the \$GET is done, if this is a non-terminal device. This
197 0197 1 | is a requested behavior change that will cause input prompts to
198 0198 1 | appear in batch log files. RMS is making a change concurrently
199 0199 1 | that will cause the actual input provided to appear in the batch
200 0200 1 | log as well, thus making a batch log an exact duplicate of how
201 0201 1 | it would appear if run interactively. MDL 26-Jul-1983
202 0202 1 | 1-093 - check for RMSS_CONTROLC completion status; call new CTRL/C signaller
203 0203 1 | if this status is returned. this change is coordinated with rev.
204 0204 1 | 2-003 of BAS\$CTRLC. MDL 12-Mar-1984
205 0205 1 | 1-094 - for the special case of channel 0, edit 1-092 should reach thru the
206 0206 1 | buddy ptr and use the output side of channel 0 to write out the
207 0207 1 | prompt string. MDL 22-Mar-1984
208 0208 1 | 1-095 - routines that set and reset record options should reset them BEFORE
209 0209 1 | calling error routines, so that subsequent I/O on the channel will
210 0210 1 | work properly (if the user handles the error). MDL 23-Mar-1984
211 0211 1 |--
212 0212 1 |
213 0213 1 | !<BLF/PAGE>

```

215 0214 1 | SWITCHES:
216 0215 1 | .SWITCHES:
217 0216 1 | .SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
218 0217 1 |
219 0218 1 | LINKAGES
220 0219 1 |
221 0220 1 |
222 0221 1 | REQUIRE 'RTLIN:OTSLNK';                                ! define all linkages
223 0222 1 |
224 0223 1 | TABLE OF CONTENTS:
225 0224 1 |
226 0225 1 | FORWARD ROUTINE
227 0226 1 | BASSWAIT : NOVALUE,
228 0227 1 | BASS$READ_WAIT,
229 0228 1 | BASSRECOUNT,
230 0229 1 | BASS$RECOU_INIT : NOVALUE,
231 0230 1 | BASS$BLNK_LINE : CALL_CCB NOVALUE,
232 0231 1 | ! write sequential list-directed
233 0232 1 | BASS$REC_WSL0 : JSB_REC NOVALUE,
234 0233 1 | BASS$REC_WSL1 : JSB_REC WSL1 NOVALUE,
235 0234 1 | BASS$REC_WSL9 : JSB_REC9 NOVALUE,
236 0235 1 | ! Mat Input
237 0236 1 | BASS$REC_MLI1 : JSB_REC1,
238 0237 1 | ! Mat Read
239 0238 1 | BASS$REC_MRE1 : JSB_REC1,
240 0239 1 | ! Mat Print
241 0240 1 | BASS$REC_MPR1 : JSB_REC1 NOVALUE,
242 0241 1 | BASS$REC_MPR9 : JSB_REC9 NOVALUE,
243 0242 1 | ! read sequential list-directed
244 0243 1 | BASS$REC_RSL0 : JSB_REC NOVALUE,
245 0244 1 | BASS$REC_RSL1 : JSB_REC1,
246 0245 1 | BASS$REC_RSL9 : JSB_REC9 NOVALUE,
247 0246 1 | ! MAT INPUT
248 0247 1 | BASS$REC_MIN0 : JSB_REC NOVALUE,
249 0248 1 | BASS$REC_MIN1 : JSB_REC1,
250 0249 1 | BASS$REC_MIN9 : JSB_REC9 NOVALUE,
251 0250 1 | ! read memory list-directed
252 0251 1 | BASS$REC_RMF0 : JSB_REC NOVALUE,
253 0252 1 | BASS$REC_RMF1 : JSB_REC1 NOVALUE,
254 0253 1 | BASS$REC_RMF9 : JSB_REC9 NOVALUE;
255 0254 1 | GLOBAL BIND
256 0255 1 | ROUTINE
257 0256 1 | ! write formatted
258 0257 1 | BASS$REC_WF0 = BASS$REC_WSL0,
259 0258 1 | BASS$REC_WF1 = BASS$REC_WSL1;
260 0259 1 | BASS$REC_WF9 = BASS$REC_WSL9;
261 0260 1 | FORWARD ROUTINE
262 0261 1 | ! record operations
263 0262 1 | BASS$REC_GSE : JSB_DO_READ NOVALUE,
264 0263 1 | BASS$REC_PSE : JSB_PUT NOVALUE,
265 0264 1 | ! GET sequential
266 0265 1 | ! PUT sequential
267 0266 1 |
268 0267 1 |
269 0268 1 |
270 0269 1 |
271 0270 1 |

```

```

272      0699 1 BASS$REC_FSE : JSB REC2 NOVALUE,
273      0700 1 BASS$REC_FRFA: JSB REC2 NOVALUE,
274      0701 1 BASS$REC_DSE : JSB RECO NOVALUE,
275      0702 1 BASS$REC_UPD : JSB DO WRITE NOVALUE,
276      0703 1 BASS$REC_RSE : JSB RECO NOVALUE,
277      0704 1 BASS$REC_SSE : JSB RECO NOVALUE,
278      0705 1 BASS$REC_PRE : JSB PUT NOVALUE,
279      0706 1 BASS$REC_GRE : JSB DO READ NOVALUE,
280      0707 1 BASS$REC_GRFA: JSB DO READ NOVALUE,
281      0708 1 BASS$REC_FRE : JSB RECO NOVALUE,
282      0709 1 BASS$REC_UNL : JSB RECO NOVALUE,
283      0710 1 BASS$REC_FEE : JSB RECO NOVALUE,
284      0711 1 BASS$REC_GIN : JSB REC_IND1 NOVALUE,
285      0712 1 BASS$REC_FIN : JSB REC_IND1 NOVALUE,
286      0713 1 BASS$REC_RIN : JSB REC_IND NOVALUE,
287      0714 1 PUT_ERROR : CALL_CCB NOVALUE,
288      0715 1 GET_ERROR : CALL_CCB NOVALUE;
289      0716 1
290      0717 1 ! INCLUDE FILES:
291      0718 1
292      0719 1
293      0720 1
294      0721 1 REQUIRE 'RTLIN:BASIOERR';           ! I/O error codes
295      0774 1
296      0775 1 REQUIRE 'RTLIN:BASFRAME';        ! Basic frame offsets
297      0978 1
298      0979 1 REQUIRE 'RTLML:OTSIISB';         ! I/O statement block (ISB) offsets
299      1147 1
300      1148 1 REQUIRE 'RTLML:OTSLUB';         ! Logical unit block (LUB) offsets
301      1288 1
302      1289 1 REQUIRE 'RTLIN:OTSMAC';         ! Macros
303      1483 1
304      1484 1 REQUIRE 'RTLIN:RTLPSECT';       ! Define DECLARE_PSECTS macro
305      1579 1
306      1580 1 REQUIRE 'RTLML:BASPAR';         ! BASIC inter-module parameters
307      1602 1
308      1603 1 LIBRARY 'RTLSTARLE';          ! STARLET library for macros and symbols
309      1604 1
310      1605 1
311      1606 1 ! MACROS:
312      1607 1
313      1608 1     NONE
314      1609 1
315      1610 1 ! EQUATED SYMBOLS:
316      1611 1
317      1612 1
318      1613 1 ! LITERAL
319      1614 1     K_MAT_CONT_CHAR = %X'26';
320      1615 1
321      1616 1     K_STOP = 0,
322      1617 1     K_SIGNAL = 1;
323      1618 1
324      1619 1
325      1620 1 ! PSECT DECLARATIONS:
326      1621 1
327      1622 1     DECLARE_PSECTS (BAS);
328      1623 1

```

! '26' - Mat Input continuation character
 ! stop after signalling this error signal and allow restart

! declare PSECTS for BASS\$ facility

```
329      1624 1 ! OWN STORAGE:  
330      1625 1 OWN  
331      1626 1 RECOUNT : INITIAL (0),  
332      1627 1 WAIT   : WORD INITIAL (0);  
333  
334  
335      1630 1 ! EXTERNAL REFERENCES:  
336      1631 1 !  
337      1632 1 !  
338      1633 1 !  
339      1634 1 EXTERNAL ROUTINE  
340      1635 1 BAS$$SIGNAL : NOVALUE,  
341      1636 1 BAS$$STOP IO : NOVALUE,  
342      1637 1 BAS$$SIGNAL_IO : NOVALUE,  
343      1638 1 BAS$$SIGNAL_CTRLC : NOVALUE;  
344  
345      1640 1 !  
346      1641 1 !  
347      1642 1 EXTERNAL LITERAL  
348      1643 1 BAS$K_OUTOF DAT : UNSIGNED (8),  
349      1644 1 BAS$K_ENDFILEDEV : UNSIGNED (8),  
350      1645 1 BAS$K_NOTENO DAT : UNSIGNED (8),  
351      1646 1 BAS$K_TOOLITDAT : UNSIGNED (8),  
352      1647 1 BAS$K_RECFILETOO : UNSIGNED (8);  
353  
354      1648 1 !  
355      1649 1 !  
1650 1 !
```

```

357 1651 1 GLOBAL ROUTINE BASS$WAIT (
358 1652 1     TIME
359 1653 1     ) : NOVALUE =
360 1654 1
361 1655 1 ++
362 1656 1     FUNCTIONAL DESCRIPTION:
363 1657 1
364 1658 1     Limits the time any input I/O statement ( INPUT, INPUT LINE, LINPUT,
365 1659 1     MAT 'all above', GET ) to any terminal will wait. If the user does not
366 1660 1     reply before the indicated number of seconds an error trap ( which the
367 1661 1     user can intercept ) will be taken. WAIT is a module level OWN in this
368 1662 1     module.
369 1663 1
370 1664 1
371 1665 1     FORMAL PARAMETERS:
372 1666 1
373 1667 1     TIME.rl.v      Number of seconds to wait, max.
374 1668 1
375 1669 1
376 1670 1     IMPLICIT INPUTS:
377 1671 1     The module level OWN WAIT
378 1672 1
379 1673 1     IMPLICIT OUTPUTS:
380 1674 1
381 1675 1     Writes to the module level OWN WAIT the number of seconds given
382 1676 1
383 1677 1     ROUTINE VALUE:
384 1678 1
385 1679 1     None
386 1680 1
387 1681 1     SIDE EFFECTS:
388 1682 1
389 1683 1     None
390 1684 1
391 1685 1     --
392 1686 1
393 1687 2     BEGIN
394 1688 2
395 1689 2     If the WAIT time is unreasonable then force it to the acceptable range.
396 1690 2     This is until the correct error message is cooked up for this error.
397 1691 2     WAIT is a module level OWN.
398 1692 2
399 1693 2     WAIT = MIN ( ABS(.TIME) , 255 );
400 1694 2     RETURN;
401 1695 2     END;

```

!Limit input wait time
!Seconds to limit time

!End of BASS\$WAIT

.TITLE BASS\$REC_PROC
.IDENT \1-095\

.PSECT _BASS\$DATA,NOEXE, PIC,2

00000000 0000 RECOUNT:.LONG 0
0000 0004 WAIT:.WORD 0

.EXTRN BASS\$SIGNAL, BASS\$STOP IO
.EXTRN BASS\$SIGNAL_IO, BASS\$SIGNAL_CTRLC

.EXTRN BASSK_OUTOF DAT
.EXTRN BASSK_ENDFILEDEV
.EXTRN BASSK_NOTENODAT
.EXTRN BASSK_TOOLITDAT
.EXTRN BASSK_RECFILETOO

.PSELECT _BASS\$CODE,NOWRT, SHR, PIC,2

```
.ENTRY BASSWAIT, Save nothing
MOVL TIME, R0
BGEQ 1$ 
MNEGL R0, R0
CMPL R0, #255
BLEQ 2$ 
MOVZBL #255, R0
MOVW R0, WAIT
RET
```

; Routine Size: 32 bytes, Routine Base: _BASS\$CODE + 0000

402 1696 1

```

: 404      1697 1 ROUTINE BASS$READ_WAIT          !Read the module level OWN WAIT
: 405      1698 1 :
: 406      1699 1
: 407      1700 1 ++
: 408      1701 1 FUNCTIONAL DESCRIPTION:
: 409      1702 1
: 410      1703 1     Read the module level OWN WAIT and return it. The value of this
: 411      1704 1     function is the current contents of wait. All routines that need
: 412      1705 1     this value must call this routine.
: 413      1706 1
: 414      1707 1
: 415      1708 1 FORMAL PARAMETERS:
: 416      1709 1     NONE
: 417      1710 1
: 418      1711 1
: 419      1712 1 IMPLICIT INPUTS:
: 420      1713 1     Reads the module level OWN WAIT
: 421      1714 1
: 422      1715 1
: 423      1716 1 IMPLICIT OUTPUTS:
: 424      1717 1     None
: 425      1718 1
: 426      1719 1
: 427      1720 1 ROUTINE VALUE:
: 428      1721 1     The contents of module level OWN WAIT
: 429      1722 1
: 430      1723 1
: 431      1724 1 SIDE EFFECTS:
: 432      1725 1     None
: 433      1726 1
: 434      1727 1 --
: 435      1728 1
: 436      1729 2 BEGIN
: 437      1730 2
: 438      1731 2 Just return the value of the module level OWN WAIT
: 439      1732 2
: 440      1733 2 RETURN .WAIT;
: 441      1734 1 END;           !End of BASS$READ_WAIT

```

0000 00000 BASS\$READ_WAIT:
 50 00000000' EF 3C 00002 .WORD Save nothing
 04 00009 MOVZWL WAIT, R0
 RET

: Routine Size: 10 bytes, Routine Base: _BASS\$CODE + 0020

: 442 1735 1

: 1697
: 1733
: 1734

```

: 444      1736 1 GLOBAL ROUTINE BASS$RECOUNT           ! RECOUNT
: 445      1737 1 : =
: 446      1738 1
: 447      1739 1 ++
: 448      1740 1 FUNCTIONAL DESCRIPTION:
: 449      1741 1
: 450      1742 1 This routine supports the Basic RECOUNT function. It returns the number
: 451      1743 1 of bytes read on the last Get. It utilizes a piece of OWN storage which
: 452      1744 1 is written to by the record processing levels which do Gets. In order
: 453      1745 1 to keep the OWN storage from having to be global, this routine is included
: 454      1746 1 in this module.
: 455      1747 1
: 456      1748 1 FORMAL PARAMETERS:
: 457      1749 1
: 458      1750 1     NONE
: 459      1751 1
: 460      1752 1 IMPLICIT INPUTS:
: 461      1753 1
: 462      1754 1     RECOUNT.rl          The number of bytes read on the last GET
: 463      1755 1
: 464      1756 1 IMPLICIT OUTPUTS:
: 465      1757 1
: 466      1758 1     NONE
: 467      1759 1
: 468      1760 1 ROUTINE VALUE:
: 469      1761 1
: 470      1762 1     NUM_OF_BYTES.wl.v    number of bytes read on last Get
: 471      1763 1
: 472      1764 1 SIDE EFFECTS:
: 473      1765 1
: 474      1766 1 --
: 475      1767 1
: 476      1768 2 BEGIN
: 477      1769 2 RETURN .RECOUNT
: 478      1770 1 END;                      ! End of BASS$RECOUNT

```

| | |
|---------------------------------------|---------------------------------------------------------------|
| 50 00000000' EF 0000 0000 04 00009 | .ENTRY BASS\$RECOUNT, Save nothing MOVL RECOUNT, R0 RET |
|---------------------------------------|---------------------------------------------------------------|

; Routine Size: 10 bytes, Routine Base: _BASS\$CODE + 002A

; 479 1771 1

: 1736
: 1769
: 1770

```
481      1772 1 GLOBAL ROUTINE BASS$RECOU_INIT : NOVALUE =      ! Initialize RECOUNT
482      1773 1
483      1774 1 ++
484      1775 1 FUNCTIONAL DESCRIPTION:
485      1776 1
486      1777 1
487      1778 1 This routine initializes the RECOUNT variable. It is used before a RUN
488      1779 1 compiler command in case the previous run of the user's program left
489      1780 1 something in RECOUNT.
490      1781 1
491      1782 1 FORMAL PARAMETERS:
492      1783 1
493      1784 1     NONE
494      1785 1
495      1786 1 IMPLICIT INPUTS:
496      1787 1
497      1788 1     NONE
498      1789 1
499      1790 1 IMPLICIT OUTPUTS:
500      1791 1
501      1792 1     RECOUNT.wl    Always set to zero.
502      1793 1
503      1794 1 ROUTINE VALUE:
504      1795 1
505      1796 1     NONE
506      1797 1
507      1798 1 SIDE EFFECTS:
508      1799 1
509      1800 1
510      1801 1
511      1802 2 BEGIN
512      1803 2     RECOUNT = 0;
513      1804 1     END;                                ! End of BASS$RECOU_INIT
```

00000000' EF 0000 00000
 CLRL RECOUNT
 RET

: Routine Size: 9 bytes, Routine Base: _BASS\$CODE + 0034

: 514 1805 1

: 1772
: 1803
: 1804

```
516      1806 1 GLOBAL ROUTINE BASS$BLNK_LINE (
517          1807 1     FORMAT_CHAR) : CALL_CCB NOVALUE =
518          1808 1                                     ! write a blank line
519          1809 1
520          1810 1 ++ FUNCTIONAL DESCRIPTION:
521          1811 1
522          1812 1     Print out a blank line. This is needed between arrays.
523          1813 1
524          1814 1 FORMAL PARAMETERS:
525          1815 1
526          1816 1     FORMAT_CHAR.rlu.v           the format character last used
527          1817 1
528          1818 1 IMPLICIT INPUTS:
529          1819 1     NONE
530          1820 1
531          1821 1 IMPLICIT OUTPUTS:
532          1822 1     NONE
533          1823 1
534          1824 1 COMPLETION CODES:
535          1825 1     NONE
536          1826 1
537          1827 1 SIDE EFFECTS:
538          1828 1     NONE
539          1829 1
540          1830 1
541          1831 1
542          1832 1     NONE
543          1833 1
544          1834 1
545          1835 1
546          1836 2 -- BEGIN
547          1837 2
548          1838 2 EXTERNAL REGISTER
549          1839 2     CCB : REF BLOCK [, BYTE];
550          1840 2
551          1841 2 LOCAL
552          1842 2     RMS_STATUS;
553          1843 2
554          1844 2
555          1845 2 + Actually put out the blank line here.
556          1846 2
557          1847 2 - CCB [RAB$W_RSZ] = 0;
558          1848 2     CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_BEG];
559          1849 2     CCB [LUB$B_BAS_VFC1] = BASSK_LF;
560          1850 2     CCB [LUB$B_BAS_VFC2] = BASSK_CR;
561          1851 2
562          1852 2     RMS_STATUS = $PUT (RAB = .CCB);
563          1853 2
564          1854 2 IF .RMS_STATUS EQL RMSS_CONTROLC
565          1855 2 THEN
566          1856 2     BAS$SIGNAL_CTRLC ();
567          1857 2
568          1858 2 IF NOT .RMS_STATUS
569          1859 2 THEN
570          1860 2     PUT_ERROR (K_STOP);
571          1861 2
572          1862 2 RETURN;
```

: 573 1863 1 END;

!End of BASS\$BLNK_LINE

| | | | | .EXTRN SYSSPUT | |
|--------------|----|-------|-------------|---------------------------------|--------|
| B0 AB | AB | 22 BC | 0004 00000 | .ENTRY BASS\$BLNK_LINE, Save R2 | ; 1806 |
| DA AB | | 8D01 | AB D0 00002 | CLRW 34(CC(B) | ; 1847 |
| | | | 8F B0 00005 | MOVL -68(CC(B), -80(CC(B) | ; 1848 |
| | | | 5B DD 00010 | MOVW #36097, -38(CC(B) | ; 1849 |
| 00000000G 00 | | | 01 FB 00012 | PUSHL CCB | ; 1852 |
| 52 | | | 50 D0 00019 | CALLS #1, SYSSPUT | |
| 00010651 8F | | | 52 D1 0001C | MOVL R0, RMS_STATUS | |
| | | | 07 12 00023 | CMPL RMS_STATUS, #67153 | 1854 |
| 00000000G 00 | | | 00 FB 00025 | BNEQ 1\$ | |
| 07 | | | 52 E8 0002C | CALLS #0, BASS\$SIGNAL_CTRLC | 1856 |
| 0000V CF | | | 7E D4 0002F | 1\$: BLBS RMS_STATUS, 2\$ | 1858 |
| | | | 01 FB 00031 | CLRL -(SP) | 1860 |
| | | | 04 00036 | CALLS #1, PUT_ERROR | |
| | | | 2\$: | RET | ; 1863 |

: Routine Size: 55 bytes, Routine Base: _BASS\$CODE + 003D

: 574 1864 1

576 1865 1 GLOBAL ROUTINE BASS\$REC_MPR1 ! Write Mat Print record
577 1866 1 : JSB_REC1 NOVALUE =
578 1867 1
579 1868 1 ++
580 1869 1 FUNCTIONAL DESCRIPTION:
581 1870 1
582 1871 1 Write one sequential formatted record and initialize for the next
583 1872 1 BASS\$REC_MPR1 writes one record for 10 MAT PRINT A() and then
584 1873 1 initializes the output buffer and returns start and end+1 of user
585 1874 1 part of record buffer to be filled by caller.
586 1875 1 FLR records are space padded.
587 1876 1
588 1877 1 FORMAL PARAMETERS:
589 1878 1
590 1879 1 NONE
591 1880 1
592 1881 1 IMPLICIT INPUTS:
593 1882 1
594 1883 1 LUB\$V_FORM_CHAR =1, comma or semicolon format character
595 1884 1 LUB\$W_RBUF_SIZE Size (bytes) allocated for record buffer at OPEN.
596 1885 1 LUB\$A_RBUF_ADR Address of record buffer from OPEN
597 1886 1 LUB\$A_BUF_END points to last char inserted into buffer
598 1887 1 by UDF level I/O.
599 1888 1 LUB\$V_FORCEABLE Indicates a forcible device
600 1889 1 LUB\$V_OUTBUF_DR Indicates that there is valid data in the output
601 1890 1 buffer
602 1891 1 RAB\$W_RSZ Record size
603 1892 1
604 1893 1 IMPLICIT OUTPUTS:
605 1894 1
606 1895 1 LUB\$B_BAS_VFC2 'Post' carriage control for terminal devices
607 1896 1 LUB\$A_BUF_PTR Address of next char in user part
608 1897 1 of record buffer
609 1898 1 LUB\$A_BUF_END Address of last+1 char in user part
610 1899 1 of record buffer
611 1900 1 LUB\$V_OUTBUF_DR indicates valid data in the output buffer
612 1901 1 LUB\$A_BUF_BEG Beginning of the user buffer
613 1902 1 RAB\$L_RBF Pointer to the user record buffer.
614 1903 1
615 1904 1 ROUTINE VALUE:
616 1905 1
617 1906 1 NONE
618 1907 1
619 1908 1 SIDE EFFECTS:
620 1909 1
621 1910 1 NONE
622 1911 1 --
623 1912 1
624 1913 2 BEGIN
625 1914 2
626 1915 2 EXTERNAL REGISTER
627 1916 2 CCB : REF BLOCK [, BYTE];
628 1917 2
629 1918 2 LOCAL
630 1919 2 RMS_STATUS;
631 1920 2
632 1921 2 !+

```

633      1922 2      | If there is no format character, then set the 'pre' and 'post'
634      1923 2      | carriage control to delimit a record.
635      1924 2
636      1925 2
637      1926 2
638      1927 2
639      1928 3
640      1929 3      IF NOT .CCB [LUB$V_FORM_CHAR]
641      1930 3      THEN
642      1931 2          BEGIN
643      1932 2          CCB [LUB$B_BAS_VFC1] = BASSK_LF;
644      1933 2          CCB [LUB$B_BAS_VFC2] = BASSK_CR;
645      1934 2          END;
646      1935 2
647      1936 2
648      1937 2
649      1938 2
650      1939 2
651      1940 2
652      1941 2      |+
653      1942 2      | Set recordsize to actual length of record
654      1943 2
655      1944 2
656      1945 2      |+
657      1946 2      | Output buffer to RMS and check for errors
658      1947 2      | If errors, SIGNAL BASS_FATSYSIO (12='FATAL SYSTEM I/O FAILURE')
659      1948 2
660      1949 2
661      1950 2      IF .RMS_STATUS EQL RMSS_CONTROLC
662      1951 2      THEN
663      1952 2          BASS$SIGNAL_CTRLC ();
664      1953 2
665      1954 2
666      1955 2
667      1956 2
668      1957 2      |+
669      1958 2      | Not OPEN or CONNECT - RMS record operation
670      1959 2
671      1960 2      |-
672      1961 2          PUT_ERROR (K_STOP);
673      1962 2
674      1963 2      |+
675      1964 2      | Return next output buffer start and end addresses
676      1965 2
677      1966 2      |-
678      1967 2      | CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
679      1968 2      | CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
680      1969 1      | RETURN;
                           END;                                ! End of routine - BASS$REC_MPR1

```

| | | | | | | | | |
|----|-------|-------|------|----------|----------------|-----------------|--------|--------------------|
| 06 | FE DA | AB AB | 8D01 | 52 02 8F | DD 00002 00007 | BAS\$REC_MPR1:: | PUSHL | R2 |
| | | | | | | | BBS #2 | -2(CC(B), 1\$ |
| | | | | | | | MOVW | #36097, -38(CC(B)) |

: 1865
: 1926
: 1929

| | | | | | | |
|--------------|-------|----|----------------|------------|--------------------------------|------|
| 22 AB | B0 AB | BC | AB A3 0000D | 1\$: SUBW3 | -68(CC(B), -80(CC(B), 34(CC(B) | 1937 |
| 28 AB | | EC | AB D0 00014 | MOVL | -20(CC(B), 40(CC(B) | 1944 |
| FE AB | | | 08 8A 00019 | BICB2 | #8, -2(CC(B) | 1945 |
| 00000000G 00 | | | 5B DD 0001D | PUSHL | CC(B | 1947 |
| 52 | | | 01 FB 0001F | CALLS | #1, SYSSPUT | |
| 00010651 8F | | | 50 D0 00026 | MOVL | R0, RMS STATUS | |
| | | | 52 D1 00029 | CMPBL | RMS_STATUS, #67153 | 1949 |
| 00000000G 00 | | | 07 12 00030 | BNEQ | 2\$ | |
| 07 | | | 00 FB 00032 | CALLS | #0, BASS\$SIGNAL_CTRLC | 1951 |
| 0000V CF | | | 52 E8 00039 | BLBS | RM\$ STATUS, 3\$ | 1953 |
| B0 AB | | EC | 7E D4 0003C | CLRL | -(SP) | 1960 |
| 50 | | | 01 FB 0003E | CALLS | #1 PUT ERROR | |
| B4 AB | | | D2 AB 3C 00043 | MOVL | -20(CC(B), -80(CC(B) | 1966 |
| | | | BB40 9E 0004C | MOVZWL | -46(CC(B), R0 | 1967 |
| | | | 04 BA 00052 | MOVAB | a-20(CC(B)[R0], -76(CC(B) | |
| | | | 05 00054 | POPR | #^M<R2> | |
| | | | | RSB | | 1969 |

; Routine Size: 85 bytes, Routine Base: _BASS\$CODE + 0074

; 681 1970 1

```

683 1971 1 GLOBAL ROUTINE BASS$REC_MPR9           ! Mat Write sequential
684 1972 1 : JSB_REC9 NOVALUE =
685 1973 1
686 1974 1
687 1975 1 ++ FUNCTIONAL DESCRIPTION:
688 1976 1
689 1977 1 This routine does not write a record. Presumably the MAT PRINT element
690 1978 1 transmitter took care of all of that. Since we do not want a blank line
691 1979 1 after the array, there is no need to write anything here.
692 1980 1
693 1981 1 FORMAL PARAMETERS:
694 1982 1
695 1983 1     NONE
696 1984 1
697 1985 1
698 1986 1
699 1987 1 IMPLICIT INPUTS:
700 1988 1     LUB$W_RBUF_SIZE      Size (bytes) allocated for record buffer at OPEN.
701 1989 1     LUB$A_RBUF_ADR       Address of record buffer from OPEN
702 1990 1
703 1991 1 IMPLICIT OUTPUTS:
704 1992 1     LUB$A_BUF_PTR        Address of next char in user part
705 1993 1             of record buffer
706 1994 1     LUB$A_BUF_END        Address of last+1 char in user part
707 1995 1             of record buffer
708 1996 1
709 1997 1 ROUTINE VALUE:
710 1998 1
711 1999 1     NONE
712 2000 1
713 2001 1 SIDE EFFECTS:
714 2002 1
715 2003 1     NONE
716 2004 1
717 2005 1
718 2006 2 BEGIN
719 2007 2
720 2008 2 EXTERNAL REGISTER
721 2009 2     CCB : REF BLOCK [, BYTE];
722 2010 2
723 2011 2
724 2012 2     !+ Return next output buffer start and end addresses
725 2013 2     !-
726 2014 2
727 2015 2     CCB [LUB$A_BUF_PTR] = .[CCB [LUB$A_RBUF_ADR];
728 2016 2     CCB [LUB$A_BUF_END] = .[CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
729 2017 2     RETURN;
730 2018 1     END;                      ! END BASS$REC_MPR9

```

| | | |
|-------|---------------------------------|---------------------------------------------------------------|
| B0 AB | EC AB DO 00000 BASS\$REC MPR9:: | MOV _L -20(CC _B), -80(CC _B) |
| B4 AB | D2 AB 3C 00005 | MOVZWL -46(CC _B), R0 |
| | EC BB40 9E 00009 | MOVAB a-20(CC _B)[R0], -76(CC _B) |

2015
2016

BASS\$REC_PROC
1-095

L 5
16-Sep-1984 01:01:12
14-Sep-1984 11:56:35

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASREC(PO.B32;1

Page 19
(9)

05 0000F RSB

; 2018

; Routine Size: 16 bytes, Routine Base: _BASS\$CODE + 00C9

; 731 2019 1

```
; 733 2020 1 GLOBAL ROUTINE BASS$REC_WSL9           ! Write sequential formatted
; 734 2021 1 : JSB_REC9 NOVALUE =
; 735 2022 1
; 736 2023 1 ++
; 737 2024 1 FUNCTIONAL DESCRIPTION:
; 738 2025 1
; 739 2026 1 Write one sequential formatted record and initialize for the next
; 740 2027 1 BASS$REC_WSF1 (and BASS$REC_WSL9) writes one output buffer and then
; 741 2028 1 initializes the output buffer and returns start and end+1 of user
; 742 2029 1 part of record buffer to be filled by caller.
; 743 2030 1 FLR records are space padded.
; 744 2031 1 /logical record number is incremented/.
; 745 2032 1
; 746 2033 1 FORMAL PARAMETERS:
; 747 2034 1
; 748 2035 1     NONE
; 749 2036 1
; 750 2037 1 IMPLICIT INPUTS:
; 751 2038 1
; 752 2039 1     ISBSV_PRINT_INI      flag to indicate whether there was an element
; 753 2040 1     transmitter
; 754 2041 1     LUBSW_RBUF_SIZE    Size (bytes) allocated for record buffer at OPEN.
; 755 2042 1     LUBSA_RBUF_ADR     Address of record buffer from OPEN
; 756 2043 1     LUBSA_BUF_END      points to last char inserted into buffer
; 757 2044 1
; 758 2045 1     LUB$V_FORM_CHAR    by UDF level I/O.
; 759 2046 1
; 760 2047 1     LUB$V_FORCEABLE   The last element transmitter ended in a comma
; 761 2048 1     LUB$V_OUTBUF_DR   or semicolon format char.
; 762 2049 1
; 763 2050 1     RABSW_RSZ        Indicates a forcible device
; 764 2051 1
; 765 2052 1     RABSL_RBF        Indicates that there is valid data in the output
; 766 2053 1     RABSL_RBF        buffer
; 767 2054 1
; 768 2055 1     ISBSV_PRINT_INI  Record size
; 769 2056 1     LUBSB_BAS_VFC2   reset flag
; 770 2057 1     LUBSA_BUF_PTR    'Post' carriage control for terminal devices
; 771 2058 1     LUBSA_BUF_END    Address of next char in user part
; 772 2059 1
; 773 2060 1     LUB$V_OUTBUF_DR of record buffer
; 774 2061 1     LUBSA_BUF_BEG   Address of last+1 char in user part
; 775 2062 1     RABSL_RBF       of record buffer
; 776 2063 1
; 777 2064 1     RABSL_RBF       indicates valid data in the output buffer
; 778 2065 1
; 779 2066 1     RABSL_RBF       Beginning of the user buffer
; 780 2067 1
; 781 2068 1     RABSL_RBF       Pointer to the user record buffer.
; 782 2069 1
; 783 2070 1
; 784 2071 1     --
; 785 2072 1
; 786 2073 2     BEGIN
; 787 2074 2
; 788 2075 2     EXTERNAL REGISTER
; ; 789 2076 2     CCB : REF BLOCK [, BYTE];
```

```
790      2077 2
791      2078 2
792      2079 2
793      2080 2
794      2081 2
795      2082 2
796      2083 2
797      2084 2
798      2085 2
799      2086 2
800      2087 2
801      2088 2
802      2089 2
803      2090 2
804      2091 2
805      2092 2
806      2093 2
807      2094 2
808      2095 2
809      2096 2
810      2097 2
811      2098 2
812      2099 2
813      2100 2
814      2101 2
815      2102 2
816      2103 2
817      2104 2
818      2105 2
819      2106 2
820      2107 2
821      2108 2
822      2109 2
823      2110 2
824      2111 2
825      2112 2
826      2113 2
827      2114 2
828      2115 2
829      2116 2
830      2117 2
831      2118 2
832      2119 2
833      2120 2
834      2121 2
835      2122 2
836      2123 2
837      2124 2
838      2125 2
839      2126 2
840      2127 2
841      2128 2
842      2129 2
843      2130 2
844      2131 2
845      2132 2
; 846      2133 2

    LOCAL
        RMS_STATUS;

    [+]
    If last element ended with a format character and not a terminal device
    then return to caller without writing anything. With CR format, we must
    PUT a whole record.
    [-]

    IF .CCB [LUB$V_FORM_CHAR] AND NOT .CCB [LUB$V_FORCIBLE] THEN RETURN;

    [+]
    Set the 'post' carriage control to carriage return
    if the last element transmitter had no format character following.
    [-]

    IF NOT .CCB [LUB$V_FORM_CHAR] THEN CCB [LUB$B_BAS_VFC2] = BASSK_CR;

    [+]
    Set recordsize to actual length of record
    [-]

    CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_BUF_BEG];

    [+]
    Output buffer to RMS and check for errors
    If errors, SIGNAL BAS$FATSYSIO (12='FATAL SYSTEM I/O FAILURE')
    [-]

    CCB [RAB$L_RBF] = .CCB [LUB$A_BUF_BEG];

    [+]
    Write something if there is something in the buffer or if there was no
    element transmitter.
    [-]

    IF .CCB [LUB$V_OUTBUF_DR] OR .CCB [ISB$V_PRINT_INI]
    THEN

        RMS_STATUS = $PUT (RAB = .CCB);

        IF .RMS_STATUS EQL RMSS_CONTROLC
        THEN
            BAS$$SIGNAL_CTRLC ();

        IF NOT .RMS_STATUS
        THEN
            PUT_ERROR (K_STOP);

        CCB [LUB$V_OUTBUF_DR] = 0;
        CCB [ISB$V_PRINT_INI] = 0;

    [+]
    If there is no format character then set the 'pre' carriage control to LF
    for the next record. This is recursive I/O and the rest of the list when
    we return should be written on the next line.
    [-]
```

```

: 847      2134 2    IF NOT .CCB [LUB$V_FORM_CHAR] THEN CCB [LUB$B_BAS_VFC1] = BASSK_LF;
: 848      2135 2
: 849      2136 2
: 850      2137 2    !+
: 851      2138 2    | Return next output buffer start and end addresses
: 852      2139 2    !-
: 853      2140 2    CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
: 854      2141 2    CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
: 855      2142 2    RETURN;
: 856      2143 1    END;                                ! END OF ROUTINE

```

| 52 DD 00000 BASS\$REC_WSL9:: | | | | | | |
|------------------------------|--------------|------|----------------|------------|--------------------------------|------|
| 0A | FE AB | | 02 E1 00002 | PUSHL | R2 | 2020 |
| 66 | FE AB | | 06 E1 00007 | BBC | #2, -2(CC(B), 1\$ | 2087 |
| 05 | FE AB | | 02 E0 0000C | BBS | #2, -2(CC(B), 2\$ | 2094 |
| 22 | DB AB | 8D | 8F 90 00011 | 1\$: MOVB | #-115, -37(CC(B)) | |
| | BO AB | BC | AB A3 00016 | 2\$: SUBW3 | -68(CC(B), -80(CC(B), 34(CC(B) | 2100 |
| | 28 AB | BC | AB D0 0001D | MOVL | -68(CC(B), 40(CC(B) | 2107 |
| 05 | FE AB | | 03 E0 00022 | BBS | #3, -2(CC(B), 3\$ | 2113 |
| 0C | 97 AB | | 03 E1 00027 | BBC | #3, -105(CC(B), 4\$ | |
| | | | 5B DD 0002C | 3\$: PUSHL | CC(B | 2116 |
| | 00000000G 00 | | 01 FB 0002E | CALLS | #1, SY\$PUT | |
| | 52 | | 50 D0 00035 | MOVL | R0, RMS_STATUS | |
| | 00010651 8F | | 52 D1 00038 | 4\$: CMPL | RMS_STATUS, #67153 | 2118 |
| | 00000000G 00 | | 07 12 0003F | BNEQ | 5\$ | |
| | 07 | | 00 FB 00041 | CALLS | #0, BASS\$SIGNAL_CTRL_C | 2120 |
| | 00000V CF | | 52 E8 00048 | 5\$: BLBS | RMS_STATUS, 6\$ | 2122 |
| | FE AB | | 7E D4 0004B | CLRL | -(SP) | 2124 |
| 04 | 97 AB | | 01 FB 0004D | CALLS | #1, PUT_ERROR | |
| | FE AB | | 08 8A 00052 | 6\$: BICB2 | #8, -2(CC(B) | 2126 |
| | DA AB | | 08 8A 00056 | BICB2 | #8, -105(CC(B) | 2127 |
| | BO AB | EC | 02 E0 0005A | BBS | #2, -2(CC(B), 7\$ | 2134 |
| | 50 | D2 | 01 90 0005F | MOVBL | #1, -38(CC(B) | |
| | B4 AB | BB40 | D2 AB 3C 00068 | 7\$: MOVL | -20(CC(B), -80(CC(B) | 2140 |
| | | | 04 9E 0006C | MOVZWL | -46(CC(B), R0 | 2141 |
| | | | 04 BA 00072 | MOVAB | @-20(CC(B)[R0], -76(CC(B) | |
| | | | 05 00074 | POPR | #^M<R2> | 2143 |
| | | | | RSB | | |

: Routine Size: 117 bytes. Routine Base: _BASS\$CODE + 00D9

: 857 2144 1

```
: 859      2145 1 GLOBAL ROUTINE BASS$REC_RSLO          ! Read initialization
860      2146 1 : JSB_REC0 NOVALUE =
861      2147 1
862      2148 1 ++
863      2149 1 FUNCTIONAL DESCRIPTION:
864      2150 1
865      2151 1 BASS$REC_RSFO (and BASS$REC_RSF1) reads one record if this is not a terminal.
866      2152 1 Then return start and end+1 of user
867      2153 1 part of record to be processed as input.
868      2154 1
869      2155 1 FORMAL PARAMETERS:
870      2156 1
871      2157 1     NONE
872      2158 1
873      2159 1 IMPLICIT INPUTS:
874      2160 1
875      2161 1     LUBSW_RBUF_SIZE      Size of record buffer allocated in OPEN.
876      2162 1     LUBSA_RBUF_ADR      Address of record buffer from OPEN.
877      2163 1     LUBSV_TERM_DEV      flag in LUB which indicates a terminal device.
878      2164 1     RABSW_RSZ        word in the RAB which contains the buffer size.
879      2165 1     RABSL_RBF        longword in RAB which points to the buffer.
880      2166 1     LUBSL_WAIT_TIME    Max time to wait for input, in seconds.
881      2167 1     WAIT            The module level OWN WAIT
882      2168 1
883      2169 1 IMPLICIT OUTPUTS:
884      2170 1
885      2171 1     RECOUNT          Global storage to hold number of bytes read from
886      2172 1     last Input.
887      2173 1     LUBSL_LOG_RECNO   Increment logical record number
888      2174 1     of next record to be read.
889      2175 1     LUBSA_BUF_PTR     points to first char of user part of
890      2176 1     record buffer.
891      2177 1     LUBSA_BUF_END     points to end+1 of user part of
892      2178 1     record buffer.
893      2179 1
894      2180 1 ROUTINE VALUE:
895      2181 1
896      2182 1     NONE
897      2183 1
898      2184 1 SIDE EFFECTS:
899      2185 1
900      2186 1     Reads next record from file on this logical unit.
901      2187 1     Throws away things that are pending in the Print buffer for non-terminal
902      2188 1     devices.
903      2189 1     SIGNALS RMS errors directly.
904      2190 1     SIGNALS BASS$K_TIMLIMEXC if a wait time was specified and we
905      2191 1     exceed it.
906      2192 1 ---
907      2193 1
908      2194 2 BEGIN
909      2195 2
910      2196 2 EXTERNAL REGISTER
911      2197 2     CCB : REF BLOCK [, BYTE];
912      2198 2
913      2199 2 LITERAL
914      2200 2     K_ESCAPE = %X'1B',
915      2201 2     K_CR = %X'0D';
```

```
: 916      2202  2
: 917      2203  2
: 918      2204  2
: 919      2205  2
: 920      2206  2
: 921      2207  2
: 922      2208  2
: 923      2209  2
: 924      2210  2
: 925      2211  2
: 926      2212  2
: 927      2213  2
: 928      2214  2
: 929      2215  2
: 930      2216  2
: 931      2217  2
: 932      2218  3
: 933      2219  2
: 934      2220  2
: 935      2221  2
: 936      2222  2
: 937      2223  2
: 938      2224  2
: 939      2225  2
: 940      2226  2
: 941      2227  2
: 942      2228  2
: 943      2229  2
: 944      2230  2
: 945      2231  2
: 946      2232  2
: 947      2233  2
: 948      2234  2
: 949      2235  2
: 950      2236  2
: 951      2237  2
: 952      2238  2
: 953      2239  2
: 954      2240  2
: 955      2241  3
: 956      2242  2
: 957      2243  2
: 958      2244  2
: 959      2245  2
: 960      2246  2
: 961      2247  2
: 962      2248  2
: 963      2249  2
: 964      2250  2
: 965      2251  2
: 966      2252  2
: 967      2253  2
: 968      2254  2
: 969      2255  3
: 970      2256  4
: 971      2257  4
: 972      2258  4

      LOCAL
        RMS_STATUS,
        WAIT_TIME;
          ! Current wait time

      !+ If a timeout has been specified, store information in the RAB to tell
      !+ RMS about it. If no timeout has been specified, clear the TMO bit
      !+ in case there was an earlier timeout specified.

      !-
      !+
      !+ If WAIT is zero then use the LUB's wait. This is to provide upward compatibility
      !+ i.e. existing EXE's can run with the LUB wait value in V2.2.

      WAIT_TIME = ( IF ( .WAIT EQL 0 ) THEN .CCB [ LUB$L_WAIT_TIME ] ELSE .WAIT );

      IF ( .WAIT_TIME EQL 0 )
      THEN
        CCB [RAB$V_TMO] = 0
      ELSE
        BEGIN
          CCB [RAB$B_TMO] = .WAIT_TIME;
          CCB [RAB$V_TMO] = 1;
        END;

      !+
      !+ Set the Read-no-echo RMS bit based on the user's last call to
      !+ ECHO or NOECHO.

      CCB [RAB$V_RNE] = .CCB [LUB$V_NOECHO];

      !+
      !+ Check to see if this is a terminal device. If this is NOT
      !+ a terminal then do a GET. GETs for terminals are done each time more
      !+ data are needed.
      !+ Read record into buffer using RMS and check for errors

      !-
      !+
      !+ IF ( NOT .CCB [LUB$V_TERM_DEV] OR .CCB [LUB$V_ANSI] )
      !+ THEN
      !+   BEGIN
      !+     LOCAL
      !+       TEMP_CCB : REF BLOCK [, BYTE];      ! Temporary CCB
      !+       TEMP_CCB = .CCB [LUB$A_BUDDY_PTR];
      !+
      !+       ! If there is something pending in the Print buffer, then $PUT it.
      !+       ! It cannot become a prompt, because RMS will throw away prompts
      !+       ! to disk files; therefore we must $PUT it.
      !-
      !+       IF (NOT .CCB [LUB$V_TERM_DEV]) AND .TEMP_CCB [LUB$V_OUTBUF_DR]
      !+       THEN
      !+         BEGIN
      !+           TEMP_CCB [RAB$W_RSZ] = .TEMP_CCB [LUB$A_BUF_PTR] - .TEMP_CCB [LUB$A_BUF_BEG];
      !+           TEMP_CCB [RAB$L_RBF] = .TEMP_CCB [LUB$A_BUF_BEG];
      !+
```

```
973      2259 4          RMS_STATUS = $PUT (RAB = .TEMP_CCB);
974      2260 4
975      2261 4
976      2262 4
977      2263 4
978      2264 4
979      2265 4
980      2266 4
981      2267 4
982      2268 4
983      2269 3
984      2270 3
985      2271 3
986      2272 3
987      2273 3
988      2274 3
989      2275 3
990      2276 3
991      2277 3
992      2278 3
993      2279 3
994      2280 3
995      2281 3
996      2282 3
997      2283 3
998      2284 3
999      2285 3
1000     2286 3
1001     2287 3
1002     2288 3
1003     2289 3
1004     2290 3
1005     2291 5
1006     2292 3
1007     2293 3
1008     2294 3
1009     2295 3
1010     2296 3
1011     2297 3
1012     2298 3
1013     2299 3
1014     2300 3
1015     2301 3
1016     2302 3
1017     2303 3
1018     2304 3
1019     2305 3
1020     2306 3
1021     2307 2
1022     2308 2
1023     2309 2
1024     2310 2
1025     2311 2
1026     2312 2
1027     2313 2
1028     2314 2
1029     2315 2

        RMS_STATUS = $PUT (RAB = .TEMP_CCB);
        IF .RMS_STATUS EQL RMSS_CONTROLC
        THEN
            BAS$SIGNAL_CTRLC ();
        IF NOT .RMS_STATUS
        THEN
            PUT_ERROR (K_STOP);
        END;

        TEMP_CCB [LUB$A_BUF_PTR] = .TEMP_CCB [LUB$A_BUF_BEG];
        RMS_STATUS = $GET (RAB = .CCB);
        IF .RMS_STATUS EQL RMSS_CONTROLC
        THEN
            BAS$SIGNAL_CTRLC ();
        IF NOT .RMS_STATUS
        THEN
            GET_ERROR (K_STOP);

        + Set RECOUNT to the number of bytes read
        | If the file is a terminal format file, then RECOUNT has to be
        | adjusted for the carriage control terminator. Because RMS does not return
        | a terminator for a file, we unconditionally put a CRLF on the end and
        | bump RECOUNT by 2.
        |

        RECOUNT = .CCB [RAB$W_RSZ] + (IF (.CCB [LUB$V_TERM_FOR]) AND ((.CCB [LUB$B_RAT] AND FAB$M_CR) NEQU 0
        THEN 2 ELSE 0);

        + Put the CR into the STV field since RMS doesn't
        | We should only do this if the record attributes indicate a CR format.
        |

        IF (.CCB [LUB$B_RAT] AND FAB$M_CR) NEQU 0 THEN CCB [RAB$L_STV] = 13;
        + Return start-1 and end+1 address of record just read
        |

        CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF] - 1;
        CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
        END
    ELSE
        + This is a terminal. Force a no data in the buffer condition
        | so the first GET is done on the element transmitter after the
        | Prompt (if any) is known.
        |

        BEGIN
```

```
; 1030      2316 3      CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF];  
; 1031      2317 3      CCB [LUB$A_BUF_END] = .CCB [LUB$A_BUF_PTR];  
; 1032      2318 2      END;  
; 1033      2319 2      RETURN;  
; 1034      2320 2      END;  
; 1035      2321 1      END;
```

! End of BASS\$REC_RSLO

.EXTRN SYSSGET

| | | | | OC | BB | 00000 BASS\$REC RSL0:: | | |
|--|--|----|----|-------|-----------|------------------------|--------|--------------------------------------------|
| | | | | 50 | 00000000' | EF 3C 00002 | PUSHR | #^M<R2,R3> |
| | | | | | | OC 12 00009 | MOVZWL | WAIT, R0 |
| | | | | 50 | CC | AB D0 0000B | BNEQ | 1\$ |
| | | | | 07 | AB | 06 12 0000F | MOVL | -52(CCB), WAIT_TIME |
| | | | | | | 02 8A 00011 | BNEQ | 1\$ |
| | | | | | | 08 11 00015 | BICB2 | #2, 7(CCB) |
| | | | | 1F | AB | 50 90 00017 | BRB | 2\$ |
| | | | | 07 | AB | 1\$: 02 88 0001B | MOVB | WAIT TIME, 31(CCB) |
| | | 01 | AB | 00 | A0 | 2\$: AB F0 0001F | BISB2 | #2, 7(CCB) |
| | | 08 | AB | FE | | 5\$: 05 E1 00026 | INSV | -96(CCB), #0, #1, 7(CCB) |
| | | 03 | AB | A1 | | 6\$: 04 E0 0002B | BBC | #5, -2(CCB), 3\$ |
| | | | | | | 7\$: 31 00030 | BBS | #4, -95(CCB), 3\$ |
| | | | | | | | BRW | 11\$ |
| | | | | 37 | AB | 8\$: B8 AB D0 00033 | MOVL | -72(CCB), TEMP CCB |
| | | 22 | A2 | 32 | AB | 9\$: 05 E0 00037 | BBS | #5, -2(CCB), 5\$ |
| | | | | | A2 | 0\$: 03 E1 0003C | BBC | #3, -2(TEMP_CCB), 5\$ |
| | | | | | | 1\$: BC A2 A3 00041 | SUBW3 | -68(TEMP_CCB), -80(TEMP_CCB), 34(TEMP_CCB) |
| | | | | | | 2\$: BC A2 D0 00048 | MOVL | -68(TEMP_CCB), 40(TEMP_CCB) |
| | | | | | | 3\$: 52 DD 0004D | PUSHL | TEMP_CCB |
| | | | | | | | CALLS | #1, SYSSPUT |
| | | | | | | | MOV | R0, RMS STATUS |
| | | | | | | | CMPL | RMS_STATUS, #67153 |
| | | | | | | | BNEQ | 4\$ |
| | | | | | | | CALLS | #0, BASS\$SIGNAL_CTRLC |
| | | | | | | | MOV | RMS STATUS, 5\$ |
| | | | | | | | BLBS | -SP |
| | | | | | | | CLRL | #1, PUT_ERROR |
| | | | | | | | CALLS | -68(TEMP_CCB), -80(TEMP_CCB) |
| | | | | | | | MOV | CCB |
| | | | | | | | PUSHL | #1, SYSSGET |
| | | | | | | | CALLS | R0, RMS STATUS |
| | | | | | | | MOV | RMS_STATUS, #67153 |
| | | | | | | | CMPL | 6\$ |
| | | | | | | | BNEQ | #0, BASS\$SIGNAL_CTRLC |
| | | | | | | | BLBS | RMS_STATUS, 7\$ |
| | | | | | | | CLRL | -SP |
| | | | | | | | CALLS | #1, GET_ERROR |
| | | 0A | AB | 0000V | CF | 7\$: 01 FB 0006E | BBC | #4, -2(CCB), 8\$ |
| | | 05 | A2 | | | 8\$: BC A2 D0 00073 | BBC | #1, -10(CCB), 8\$ |
| | | | | | | 9\$: 5B DD 00078 | MOVL | #2, R0 |
| | | | | | | | BRB | 9\$ |
| | | | | | | | CLRL | R0 |
| | | | | | | | MOVZWL | 34(CCB), R1 |
| | | | | | | | ADDL3 | R1, R0, RECOUNT |
| | | | | | | | BBC | #1, -10(CCB), 10\$ |

| | | | | |
|-------|----------------|------------------------------|------------------------------|------|
| B0 AB | 0C AB | 0D D0 000C0 | 10\$: MOVL #13, 12(CCB) | |
| | 28 AB | 01 C3 000C4 | SUBL3 #1, 40(CCB), -80(CCB) | 2304 |
| | 50 | 22 AB 3C 000CA | MOVZWL 34(CCB), R0 | 2305 |
| | B4 AB | 28 BB40 9E 000CE | MOVAB @40(CCB)[R0], -76(CCB) | |
| | | 0A 11 000D4 | BRB 12\$ | 2241 |
| B0 AB | 28 AB D0 000D6 | 11\$: MOVL 40(CCB), -80(CCB) | 2316 | |
| B4 AB | B0 AB D0 000DB | MOVL -80(CCB), -76(CCB) | 2317 | |
| | OC BA 000E0 | 12\$: POPR #^M<R2,R3> | | |
| | 05 000E2 | RSB | 2321 | |

; Routine Size: 227 bytes, Routine Base: _BASS\$CODE + 014E

; 1036 2322 1

: 1038 2323 1 GLOBAL ROUTINE BASS\$REC_MINO ! MAT Input initialization
1039 2324 1 : JSB_REC0 NOVALUE =
1040 2325 1
1041 2326 1 ++
1042 2327 1 FUNCTIONAL DESCRIPTION:
1043 2328 1
1044 2329 1 BASS\$REC_RSFO (and BASS\$REC_RSF1) reads one record if this is not a terminal.
1045 2330 1 Then return start and end+1 of user
1046 2331 1 part of record to be processed as input.
1047 2332 1
1048 2333 1 FORMAL PARAMETERS:
1049 2334 1
1050 2335 1 NONE
1051 2336 1
1052 2337 1 IMPLICIT INPUTS:
1053 2338 1
1054 2339 1 LUBSW_RBUF_SIZE Size of record buffer allocated in OPEN.
1055 2340 1 LUBSA_RBUF_ADR Address of record buffer from OPEN.
1056 2341 1 LUB\$V_TERM_DEV flag in LUB which indicates a terminal device.
1057 2342 1 RAB\$W_RSZ word in the RAB which contains the buffer size.
1058 2343 1 RABSL_RBF longword in RAB which points to the buffer.
1059 2344 1 LUBSL_WAIT_TIME Max time to wait for input, in seconds.
1060 2345 1 WAIT Module level OWN WAIT
1061 2346 1
1062 2347 1 IMPLICIT OUTPUTS:
1063 2348 1
1064 2349 1 RECOUNT Global storage to hold number of bytes read from
1065 2350 1 last input.
1066 2351 1 LUB\$L_LOG_RECNO Increment logical record number
1067 2352 1 of next record to be read.
1068 2353 1 LUB\$A_BUF_PTR points to first char of user part of
1069 2354 1 record buffer.
1070 2355 1 LUB\$A_BUF_END points to end+1 of user part of
1071 2356 1 record buffer.
1072 2357 1
1073 2358 1 ROUTINE VALUE:
1074 2359 1
1075 2360 1 NONE
1076 2361 1
1077 2362 1 SIDE EFFECTS:
1078 2363 1
1079 2364 1 Reads next record from file on this logical unit.
1080 2365 1 Throws away things that are pending in the Print buffer for non-terminal
1081 2366 1 devices.
1082 2367 1 SIGNALS BASS\$FATSYSIO (12='FATAL SYSTEM I/O FAILURE')
1083 2368 1 SIGNALS BASS\$ENDFILDEV (11='END-OF-FILE ON DEVICE')
1084 2369 1 SIGNALS BASS\$RECFILTOO if record too big
1085 2370 1 SIGNALS BASS\$TIMLIMEXC if a wait time was specified and we
1086 2371 1 exceed it.
1087 2372 1 --
1088 2373 1 BEGIN
1089 2374 2
1090 2375 2 EXTERNAL REGISTER
1091 2376 2 CCB : REF BLOCK [, BYTE];
1092 2377 2
1093 2378 2
1094 2379 2 LITERAL

```
1095      2380 2      K_ESCAPE = %X'1B'.
1096      2381 2      K_R = %X'0D';
1097      2382 2
1098      2383 2      LOCAL
1099      2384 2      RMS_STATUS,
1100      2385 2      WAIT_TIME;           !Current wait time
1101      2386 2
1102      2387 2      + If a timeout has been specified, store information in the RAB to tell
1103      2388 2      RMS about it. If no timeout has been specified, clear the TMO bit
1104      2389 2      in case there was an earlier timeout specified.
1105      2390 2
1106      2391 2
1107      2392 2
1108      2393 2
1109      2394 2      + If WAIT is zero then use the LUB's wait. This is to provide upward compatibility
1110      2395 2      , i.e. existing EXE's can run with the LUB wait value in V2.2.
1111      2396 2
1112      2397 2      - WAIT_TIME = ( IF ( .WAIT EQL 0 ) THEN .CCB [ LUB$L_WAIT_TIME ] ELSE .WAIT );
1113      2398 2
1114      2399 3      IF (.WAIT_TIME EQL 0)
1115      2400 2      THEN
1116      2401 2      CCB [RAB$V_TMO] = 0
1117      2402 2      ELSE
1118      2403 2      BEGIN
1119      2404 3      CCB [RAB$B_TMO] = WAIT_TIME;
1120      2405 3      CCB [RAB$V_TMO] = 1;
1121      2406 2      END;
1122      2407 2
1123      2408 2
1124      2409 2
1125      2410 2      + Set the Read-no-echo RMS bit based on the user's last call to
1126      2411 2      ECHO or NOECHO.
1127      2412 2
1128      2413 2      CCB [RAB$V_RNE] = .CCB [LUB$V_NOECHO];
1129      2414 2
1130      2415 2
1131      2416 2      + Check to see if this is a terminal device. If this is NOT
1132      2417 2      a terminal then do a GET. GETs for terminals are done each time more
1133      2418 2      data are needed.
1134      2419 2      Read record into buffer using RMS and check for errors
1135      2420 2      If end-of-file, SIGNAL BAS$K_ENDFILDEV (11='END-OF-FILE ON DEVICE')
1136      2421 2      If record too big for record-buffer, SIGNAL BAS$K_RECVTOO.
1137      2422 2      If errors, SIGNAL BAS$K_FATSYSIO (12='FATAL SYSTEM I/O ERROR')
1138      2423 2
1139      2424 2
1140      2425 3      IF ( NOT .CCB [LUB$V_TERM_DEV])
1141      2426 2      THEN
1142      2427 3      BEGIN
1143      2428 3
1144      2429 3      LOCAL
1145      2430 3      TEMP_CCB : REF BLOCK [, BYTE];      ! Temporary CCB
1146      2431 3      TEMP_CCB = .CCB [LUB$A_BUDDY_PTR];
1147      2432 3
1148      2433 3
1149      2434 3      + If there is something pending in the Print buffer, then $PUT it.
1150      2435 3      It cannot become a prompt, because RMS will throw away prompts
1151      2436 3      to disk files; therefore we must $PUT it.
```

```
1152      2437 3
1153      2438 3
1154      2439 3
1155      2440 4
1156      2441 4
1157      2442 4
1158      2443 4
1159      2444 4
1160      2445 4
1161      2446 4
1162      2447 4
1163      2448 4
1164      2449 4
1165      2450 4
1166      2451 4
1167      2452 4
1168      2453 3
1169      2454 3
1170      2455 3
1171      2456 3
1172      2457 3
1173      2458 3
1174      2459 3
1175      2460 3
1176      2461 3
1177      2462 3
1178      2463 3
1179      2464 3
1180      2465 3
1181      2466 3
1182      2467 3
1183      2468 3
1184      2469 3
1185      2470 3
1186      2471 3
1187      2472 3
1188      2473 4
1189      2474 4
1190      2475 4
1191      2476 4
1192      2477 4
1193      2478 3
1194      2479 3
1195      2480 3
1196      2481 3
1197      2482 3
1198      2483 3
1199      2484 3
1200      2485 3
1201      2486 3
1202      2487 3
1203      2488 3
1204      2489 3
1205      2490 3
1206      2491 3
1207      2492 3
1208      2493 3

      !-
      IF (NOT .CCB [LUB$V_TERM_DEV]) AND .TEMP_CCB [LUB$V_OUTBUF_DR]
      THEN
        BEGIN
          TEMP_CCB [RAB$W_RSZ] = .TEMP_CCB [LUB$A_BUF_PTR] - .TEMP_CCB [LUB$A_BUF_BEG];
          TEMP_CCB [RAB$L_RBF] = .TEMP_CCB [LUB$A_BUF_BEG];
          RMS_STATUS = $PUT (RAB = .CCB);
          IF .RMS_STATUS EQL RMSS_CONTROLC
          THEN
            BASS$SIGNAL_CTRLC ();
          IF NOT .RMS_STATUS
          THEN
            PUT_ERROR (K_STOP);
          END;

          TEMP_CCB [LUB$A_BUF_PTR] = .TEMP_CCB [LUB$A_BUF_BEG];
          RMS_STATUS = $GET (RAB = .CCB);
          IF .RMS_STATUS EQL RMSS_CONTROLC
          THEN
            BASS$SIGNAL_CTRLC ();
          IF NOT .RMS_STATUS
          THEN
            GET_ERROR (K_STOP);

          !+
          | Set RECOUNT to the number of bytes read
          | If the file is a terminal format file, then RECOUNT has to be
          | adjusted for the carriage control terminator.
          |-
```

RECOUNT = .CCB [RAB\$W_RSZ] + (IF .CCB [LUB\$V_TERM_FOR] THEN SELECTONEU .CCB [RAB\$W_STV0] OF
SET
[K_ESCAPE] : .CCB [RAB\$W_STV2];
[K_CR] : 2;
[OTHERWISE] : 0;
TES ELSE 0);

!+
| Return start-1 and end+1 address of record just read
|-

CCB [LUB\$A_BUF_PTR] = .CCB [RAB\$L_RBF] - 1;
CCB [LUB\$A_BUF_END] = .CCB [RAB\$L_RBF] + .CCB [RAB\$W_RSZ];

!+
| Check for an '=' as the last character of the record. If it is there,
| it is a continuation character and signifies that there is more data to
| come in the next record.
|-

IF .(.CCB [LUB\$A_BUF_END] - 1)<0, 8> EQLU K_MAT_CONT_CHAR

```

1209      2494 3      THEN
1210      2495 4      BEGIN
1211      2496 4      CCB [LUB$A_BUF-END] = .CCB [LUB$A_BUF-END] - 1;
1212      2497 4      CCB [ISBSV_MAT_CONT] = 1;
1213      2498 4      END
1214      2499 3      ELSE
1215      2500 3      CCB [ISBSV_MAT_CONT] = 0;
1216      2501 3      END
1217      2502 3      ELSE
1218      2503 2
1219      2504 2
1220      2505 2
1221      2506 2      !+
1222      2507 2      This is a terminal. Force a no data in the buffer condition
1223      2508 2      so the first GET is done on the element transmitter after the
1224      2509 2      Prompt (if any) is known. Set the MAT Input continuation flag so that the element
1225      2510 2      transmitter (REC1) can read the first record.
1226      2511 2
1227      2512 3      BEGIN
1228      2513 3      CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF];
1229      2514 3      CCB [LUB$A_BUF-END] = .CCB [LUB$A_BUF_PTR];
1230      2515 3      CCB [ISBSV_MAT_CONT] = 1;
1231      2516 2      END;
1232      2517 2
1233      2518 2      RETURN;
1234      2519 1      END;

                                         ! End of BASS$REC_MINO

```

| OC BB 00000 BASS\$REC MINO:: | | | | | | |
|------------------------------|--|----------|-------------------|--------|--------------------------------------------|------|
| | | | | PUSHR | #^M<R2,R3> | 2323 |
| | | | 50 00000000' | MOVZWL | WAIT, R0 | 2397 |
| | | | EF 3C 00002 | BNEQ | 1\$ | |
| | | | 0C 12 00009 | MOVL | -52(CCB), WAIT_TIME | |
| | | | AB D0 0000B | BNEQ | 1\$ | |
| | | | 06 12 0000F | BICB2 | #2, 7(CCB) | 2399 |
| | | 07 AB | 02 8A 00011 | BRB | 2\$ | 2401 |
| | | | 08 11 00015 | MOVB | WAIT_TIME, 31(CCB) | |
| | | | 50 90 00017 | 1\$: | | |
| | | 1F AB | 02 88 0001B | BISB2 | #2, 7(CCB) | 2404 |
| | | 07 AB | A0 AB F0 0001F | INSV | -96(CCB), #0, #1, 7(CCB) | 2405 |
| | | 01 00 | 05 E1 00026 | BBC | #5, -2(CCB), 3\$ | 2413 |
| | | 03 FE AB | 00BC 31 0002B | BRW | 12\$ | 2425 |
| | | | AB D0 0002E | MOVL | -72(CCB), TEMP_CCB | 2431 |
| | | | 05 E0 00032 | BBS | #5, -2(CCB), 5\$ | 2438 |
| | | 07 52 | B8 AB D0 00032 | BBC | #3, -2(TEMP_CCB), 5\$ | |
| | | 37 FE AB | 03 E1 00037 | SUBW3 | -68(TEMP_CCB), -80(TEMP_CCB), 34(TEMP_CCB) | 2441 |
| | | 32 FE A2 | A2 A3 0003C | MOVL | -68(TEMP_CCB), 40(TEMP_CCB) | 2442 |
| | | 22 A2 | A2 D0 00043 | PUSHL | CCB | 2444 |
| | | 28 A2 | BC BC 5B DD 00048 | CALLS | #1, SY\$PUT | |
| | | | 01 FB 0004A | MOV | R0, RMS STATUS | |
| | | | 53 50 D0 00051 | CMPL | RMS_STATUS, #67153 | 2446 |
| | | | 53 D1 00054 | BNEQ | 4\$ | |
| | | | 07 12 0005B | CALLS | #0, BASS\$SIGNAL_CTRLC | 2448 |
| | | | 00 FB 0005D | BLBS | RM\$ STATUS, 5\$ | 2450 |
| | | | 53 E8 00064 | CLRL | -(SP) | 2452 |
| | | | 7E D4 00067 | | | |

; Routine Size: 251 bytes, Routine Base: _BASSCODE + 0231

; 1235 2520 1

```
1237 2521 1 GLOBAL ROUTINE BASS$REC_RSL1           ! Read element transmitter
1238 2522 1 : JSB_REC1 =
1239 2523 1
1240 2524 1 ++
1241 2525 1 FUNCTIONAL DESCRIPTION:
1242 2526 1
1243 2527 1     BASS$REC_RSL1 reads one record if this is a terminal device.
1244 2528 1     Otherwise an error is signalled.
1245 2529 1     Then return start and end+1 of user
1246 2530 1     part of record to be processed as input.
1247 2531 1
1248 2532 1 FORMAL PARAMETERS:
1249 2533 1
1250 2534 1     NONE
1251 2535 1
1252 2536 1 IMPLICIT INPUTS:
1253 2537 1
1254 2538 1     LUB$W_RBUF_SIZE      Size of record buffer allocated in OPEN.
1255 2539 1     LUB$A_RBUF_ADDR      Address of record buffer from OPEN.
1256 2540 1     LUB$V_TERM_DEV       flag indicating a terminal device.
1257 2541 1     RAB$L_RBF            Pointer to buffer
1258 2542 1     RAB$W_RSZ            buffer size
1259 2543 1
1260 2544 1 IMPLICIT OUTPUTS:
1261 2545 1
1262 2546 1     RECOUNT             Own storage for RECOUNT function.
1263 2547 1     LUB$A_BUF_PTR        points to first char of user part of
1264 2548 1
1265 2549 1     LUB$A_BUF_END         record buffer.
1266 2550 1
1267 2551 1
1268 2552 1 ROUTINE VALUE:
1269 2553 1
1270 2554 1     NONE
1271 2555 1
1272 2556 1 SIDE EFFECTS:
1273 2557 1
1274 2558 1     Reads next record from file on this logical unit.
1275 2559 1     SIGNALS Insufficient data or any resultant RMS errors.
1276 2560 1 --
1277 2561 1
1278 2562 2 BEGIN
1279 2563 2
1280 2564 2 EXTERNAL REGISTER
1281 2565 2     CCB : REF BLOCK [, BYTE];
1282 2566 2
1283 2567 2 LOCAL
1284 2568 2     RMS_STATUS,
1285 2569 2     T_CCB : REF BLOCK [, BYTE];
1286 2570 2
1287 2571 2 LITERAL
1288 2572 2     K_ESCAPE = %X'1B',
1289 2573 2     K_CR = %X'0D';
1290 2574 2
1291 2575 2
1292 2576 2     !+
1293 2577 2     | Check to see if this is a terminal device. If this is
1294 2578 2     | a terminal then do a GET. GETs for terminals are done each time more
```

```
1294      2578 2 | data are needed. If this is not a terminal device then error.  
1295      2579 2 | Read record into buffer using RMS and check for errors  
1296      2580 2 |  
1297      2581 2 |  
1298      2582 2 IF (NOT .CCB [LUB$V_ANSI]) AND .CCB [LUB$V_TERM_DEV]  
1299      2583 2 THEN  
1300      2584 3 BEGIN  
1301      2585 3 |  
1302      2586 3 RMS_STATUS = $GET (RAB = .CCB);  
1303      2587 3 |  
1304      2588 3 IF .RMS_STATUS EQL RMS$CONTROL_C  
1305      2589 3 THEN  
1306      2590 3 | BAS$$SIGNAL_CTRL_C ();  
1307      2591 3 |  
1308      2592 3 IF NOT .RMS_STATUS  
1309      2593 3 THEN  
1310      2594 3 | GET_ERROR (K_STOP);  
1311      2595 3 |  
1312      2596 3 |  
1313      2597 3 | Return start-1 and end+1 address of record just read  
1314      2598 3 | LUB$A_BUF_PTR is set to the beginning-1 of the buffer only for BASIC  
1315      2599 3 | Input. This is seen as a solution to the problem of the user entering  
1316      2600 3 | <return> as the response to a prompt (null input record) and an empty  
1317      2601 3 | or depleted buffer which requires another Get.  
1318      2602 3 | The algorithm:  
1319      2603 3 | 1) Does LUB$A_BUF_PTR = LUB$A_BUF_END?  
1320      2604 3 | | T: The buffer is depleted = another Get is required.  
1321      2605 3 | 2) Add one to LUB$A_BUF_PTR  
1322      2606 3 | 3) Does LUB$A_BUF_PTR = LUB$A_BUF_END?  
1323      2607 3 | | T: Return the default value.  
1324      2608 3 | 4) Scan for the next field.  
1325      2609 3 |  
1326      2610 3 |  
1327      2611 3 |  
1328      2612 3 | CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF] - 1;  
1329      2613 3 | CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];  
1330      2614 3 | END  
1331      2615 2 ELSE  
1332      2616 2 |  
1333      2617 2 |  
1334      2618 2 | This is not a terminal device  
1335      2619 2 | Signal insufficient data unless this is an ANSI INPUT.  
1336      2620 2 | ANSI INPUT errors should cause the statement to be restarted.  
1337      2621 2 | (This happens in BAS$$HANDLER).  
1338      2622 2 |  
1339      2623 2 |  
1340      2624 2 IF NOT .CCB [LUB$V_ANSI]  
1341      2625 2 THEN  
1342      2626 2 | BAS$$SIGNAL (BASSK_NOTENODAT)  
1343      2627 2 ELSE  
1344      2628 2 | BAS$$SIGNAL_IO (BASSK_TOOLITDAT);  
1345      2629 2 |  
1346      2630 2 |  
1347      2631 2 | Update the cursor position if this input was terminated by an escape.  
1348      2632 2 | Save cursor position if last PRINT terminator was a semi or comma.  
1349      2633 2 | Use BUDDY_PTR 'cuz we want to use the PRINT data base for channel 0  
1350      2634 2 |
```

```

: 1351      2635  2      T_CCB = .CCB [LUB$A_BUDDY_PTR];
: 1352      2636  3      T_CCB [LUB$L_PRINT_POS] = TIF .CCB [RAB$W_STV0] EQL K_ESCAPE AND .T_CCB [LUB$V_FORM_CHAR] EQLU 1
: 1353      2637  3      THEN .CCB [RAB$W_RSZ] + .T_CCB [LUB$L_PRINT_POS] + 1
: 1354      2638  2      ELSE 0;
: 1355      2639  2
: 1356      2640  2
: 1357      2641  2      !+ Set RECOUNT to the number of bytes read
: 1358      2642  2      If the file is a terminal format file, then RECOUNT has to be
: 1359      2643  2      adjusted for the carriage control terminator.
: 1360      2644  2      !-
: 1361      2645  2
: 1362      2646  3      RECOUNT = .CCB [RAB$W_RSZ] + (IF .CCB [LUB$V_TERM_FOR] THEN SELECTONEU .CCB [RAB$W_STV0] OF
: 1363      2647  3      SET
: 1364      2648  3      [K_ESCAPE] : .CCB [RAB$W_STV2];
: 1365      2649  3      [K_CR] : 2;
: 1366      2650  3      [OTHERWISE] : 0;
: 1367      2651  2      TES ELSE 0);
: 1368      2652  2      RETURN 1;
: 1369      2653  1      END;
                                         ! End of BASS$REC_RSL1

```

| | | | | 52 DD 00000 BASS\$REC_RSL1:: | | |
|-----------|----|-------|-----|------------------------------|---------------------------------------|------|
| 4F | A1 | AB | 04 | E0 00002 | PUSHL R2 | 2521 |
| 38 | FE | AB | 05 | E1 00007 | BBS #4, -95(CC(B), 4\$ | 2582 |
| | | | 5B | DD 0000C | BBC #5, -2(CC(B), 3\$ | |
| | | | 01 | FB 0000E | PUSHL CCB | 2586 |
| 00000000G | 00 | | 50 | DO 00015 | CALLS #1, SYSS\$GET | |
| 00010651 | 52 | | 52 | D1 00018 | MOVL R0, RMS_STATUS | |
| | 8F | | 07 | 12 0001F | CMPL RM\$_STATUS, #67153 | 2588 |
| 00000000G | 00 | | 00 | FB 00021 | BNEQ 1\$ | |
| | 07 | | 52 | E8 00028 | CALLS #0, BASS\$SIGNAL_CTRLC | 2590 |
| | | | 7E | D4 0002B | BLBS RM\$ STATUS, 2\$ | 2592 |
| | | | 01 | FB 0002D | CLRL -(SP) | 2594 |
| 80 | AB | 0000V | 01 | C3 00032 | CALLS #1, GET_ERROR | |
| | | 28 | AB | 22 | 2\$: SUBL3 #1, 40(CC(B), -80(CC(B) | 2612 |
| | | | 50 | 3C 00038 | MOVZWL 34(CC(B), R0 | 2613 |
| | | B4 | AB | 28 BB40 | MOVAB @40(CC(B)[R0], -76(CC(B) | |
| | | | 1D | 11 00042 | BRB 5\$ | 2582 |
| 0D | A1 | AB | 04 | E0 00044 | 3\$: BBS #4, -95(CC(B), 4\$ | 2624 |
| | | 7E | 00G | 8F 9A 00049 | MOVZBL #BASS\$K_NOTE\$DAT, -(SP) | 2626 |
| 00000000G | 00 | | 01 | FB 0004D | CALLS #1, BASS\$SIGNAL | |
| | | | 0B | 11 00054 | BRB 5\$ | 2628 |
| 00000000G | 7E | | 00G | 8F 9A 00056 | 4\$: MOVZBL #BASS\$K_TOOL\$DAT, -(SP) | |
| | 00 | | 01 | FB 0005A | CALLS #1, BASS\$SIGNAL_IO | 2635 |
| | | | 51 | AB DO 00061 | 5\$: MOVL -72(CC(B), T_CC(B | |
| | | B8 | AB | 52 | MOVZWL 12(CC(B), R2 | 2636 |
| | | | OC | 3C 00065 | CMPW R2, #27 | |
| | | 1B | 52 | B1 00069 | BNEQ 6\$ | |
| | | | 11 | 12 0006C | BBC #2, -2(T_CC(B), 6\$ | |
| OC | FE | A1 | 02 | E1 0006E | MOVZWL 34(CC(B), R0 | 2637 |
| | | 50 | 22 | AB 3C 00073 | ADDL2 -56(T_CC(B), R0 | |
| | | 50 | C8 | A1 CO 00077 | INCL R0 | |
| | | | 50 | D6 0007B | BRB 7\$ | |
| | | | 02 | 11 0007D | CLRL R0 | 2636 |
| | | | 50 | D4 0007F | 6\$: | |

C 7
16-Sep-1984 01:01:12 VAX-11 Bliss-32 v4.0-742
14-Sep-1984 11:56:35 [BASRTL.SRC]BASRECPRO.B32;1

Page 36
(13)

| | | | | | | | | | |
|--------------|----------------------|----|----------------------|----------------------------|-------------------------------------------|-----------|-----------------------------------------|-----------------------------------------------------------|----------------------|
| 15 | C8 FE AB 1B | A1 | 50 04 52 06 | D0 E1 B1 12 | 00081 00085 0008A 0008D | 7\$:: | MOVL BBC CMPW BNEQ | R0, -56(T CCB) #4, -2(CCB), 9\$ R2, #27 8\$ | 2646 2648 |
| | 50 | OE | AB 0C | 3C 11 | 0008F 00093 | | MOVZWL BRB | 14(CCB), R0 10\$ | |
| | 0D | | 52 05 | B1 12 | 00095 00098 | 8\$:: | CMPW BNEQ | R2, #13 9\$ | 2649 |
| | 50 | | 02 02 | D0 11 | 0009A 0009D | | MOVL BRB | #2, R0 10\$ | |
| | 51 | 22 | 50 51 50 | D4 3C 01 01 04 | 0009F 000A1 000A5 000AD 000B0 | 9\$:: | CLRL MOVZWL ADDL3 MOVL POPR | R0 34(CCB), R1 R1, R0, RECOUNT #1, R0 #^M<R2> | 2646 2652 2653 |
| 00000000' EF | | | | | 05 | 000B2 | RSB | | |

; Routine Size: 179 bytes, Routine Base: _BASS\$CODE + 032C

: 1370 2654 1

```

: 1372      2655 1 GLOBAL ROUTINE BASS$REC_MIN1           ! MAT Input element transmitter
: 1373      2656 1 : JSB_REC1 =
: 1374      2657 1 ++
: 1375      2658 1 FUNCTIONAL DESCRIPTION:
: 1376      2659 1
: 1377      2660 1 BASS$REC_MIN1 reads one record and checks for a continuation character.
: 1378      2661 1 Then return start and end+1 of user
: 1379      2662 1 part of record to be processed as input.
: 1380      2663 1
: 1381      2664 1 FORMAL PARAMETERS:
: 1382      2665 1
: 1383      2666 1
: 1384      2667 1     NONE
: 1385      2668 1
: 1386      2669 1 IMPLICIT INPUTS:
: 1387      2670 1
: 1388      2671 1     LUB$W_RBUF_SIZE      Size of record buffer allocated in OPEN.
: 1389      2672 1     LUB$A_RBUF_ADDR    Address of record buffer from OPEN.
: 1390      2673 1     LUB$V_TERM_DEV    flag indicating a terminal device.
: 1391      2674 1     RAB$L_RBF        Pointer to buffer
: 1392      2675 1     RAB$W_RSZ        buffer size
: 1393      2676 1
: 1394      2677 1 IMPLICIT OUTPUTS:
: 1395      2678 1
: 1396      2679 1     RECOUNT          Own storage for RECOUNT function.
: 1397      2680 1     LUB$A_BUF_PTR    points to first char of user part of
: 1398      2681 1     LUB$A_BUF_END    record buffer.
: 1399      2682 1     LUB$A_BUF_END    points to end+1 of user part of
: 1400      2683 1     record buffer.
: 1401      2684 1
: 1402      2685 1
: 1403      2686 1
: 1404      2687 1     NONE
: 1405      2688 1
: 1406      2689 1 SIDE EFFECTS:
: 1407      2690 1
: 1408      2691 1     Reads next record from file on this logical unit.
: 1409      2692 1     SIGNALS any resultant RMS errors.
: 1410      2693 1 --
: 1411      2694 1     BEGIN
: 1412      2695 2     EXTERNAL REGISTER
: 1413      2696 2     CCB : REF BLOCK [, BYTE];
: 1414      2697 2
: 1415      2698 2     LITERAL
: 1416      2699 2     K_ESCAPE = %X'1B',
: 1417      2700 2     K_CR = %X'0D';
: 1418      2701 2
: 1419      2702 2
: 1420      2703 2
: 1421      2704 2 LOCAL
: 1422      2705 2     RMS_STATUS,
: 1423      2706 2     T [B : REF BLOCK [, BYTE],
: 1424      2707 2     STATUS;           ! Return status to UDF of whether
: 1425      2708 2
: 1426      2709 2
: 1427      2710 2
: 1428      2711 2     !+           ! to keep reading

```

```

1429 2712 2 | Read record into buffer using RMS and check for errors and a continuation character
1430 2713 2 | Signal any RMS errors directly.
1431 2714 2 |
1432 2715 2
1433 2716 2 IF .CCB [ISBSV_MAT_CONT]
1434 2717 2 THEN
1435 2718 3 BEGIN
1436 2719 3
1437 2720 3 RMS_STATUS = $GET (RAB = .CCB);
1438 2721 3
1439 2722 3 IF .RMS_STATUS EQL RMSS_CONTROLC
1440 2723 3 THEN
1441 2724 3 BAS$SIGNAL_CTRLC ();
1442 2725 3
1443 2726 3 IF NOT .RMS_STATUS
1444 2727 3 THEN
1445 2728 3 GET_ERROR (K_STOP);
1446 2729 3
1447 2730 3 +
1448 2731 3 | Return start-1 and end+1 address of record just read
1449 2732 3 | LUBSA_BUF_PTR is set to the beginning-1 of the buffer only for BASIC
1450 2733 3 | Input. This is seen as a solution to the problem of the user entering
1451 2734 3 | <return> as the response to a prompt (null input record) and an empty
1452 2735 3 | or depleted buffer which requires another Get.
1453 2736 3 | The algorithm:
1454 2737 3 | 1) Does LUBSA_BUF_PTR = LUBSA_BUF_END?
1455 2738 3 | T: The buffer is depleted = another Get is required.
1456 2739 3 | 2) Add one to LUBSA_BUF_PTR
1457 2740 3 | 3) Does LUBSA_BUF_PTR = LUBSA_BUF_END?
1458 2741 3 | T: Return the default value.
1459 2742 3 | 4) Scan for the next field.
1460 2743 3
1461 2744 3
1462 2745 3
1463 2746 3 | CCB [LUBSA_BUF_PTR] = .CCB [RAB$L_RBF] - 1;
1464 2747 3 | CCB [LUBSA_BUF-END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
1465 2748 3
1466 2749 3 +
1467 2750 3 | Check for an '8' as the last character of the record. If it is there,
1468 2751 3 | it is a continuation character and signifies that there is more data to
1469 2752 3 | come in the next record.
1470 2753 3
1471 2754 3
1472 2755 3 IF .{.CCB [LUBSA_BUF-END] - 1}<0, 8> EQLU K_MAT_CONT_CHAR
1473 2756 3 THEN
1474 2757 4 BEGIN
1475 2758 4 | CCB [LUBSA_BUF-END] = .CCB [LUBSA_BUF-END] - 1;
1476 2759 4 | CCB [ISBSV_MAT_CONT] = 1;
1477 2760 4 END
1478 2761 3 ELSE
1479 2762 3 | CCB [ISBSV_MAT_CONT] = 0;
1480 2763 3
1481 2764 3 +
1482 2765 3 | Update the cursor position if this input was terminated by an escape.
1483 2766 3 | Save the cursor position if last PRINT terminator was a semi or comma.
1484 2767 3 | Use BUDDY_PTR 'cuz we want to use the PRINT data base for channel 0
1485 2768 3

```

```

1486    2769  3      T_CCB = .CCB [LUBSA_BUDDY_PTR];
1487    2770  4      T_CCB [LUB$L_PRINT_POS] = -(IF .CCB [RAB$W_STV0] EQL K_ESCAPE AND .T_CCB [LUB$V_FORM_CHAR] EQLU 1
1488    2771  4      THEN .CCB [RAB$W_RSZ] + .T_CCB [LUB$L_PRINT_POS] + T
1489    2772  3      ELSE 0);
1490    2773  3
1491    2774  3
1492    2775  3      |+
1493    2776  3      | Set RECOUNT to the number of bytes read
1494    2777  3      | If the file is a terminal format file, then RECOUNT has to be
1495    2778  3      | adjusted for the carriage control terminator.
1496    2779  3
1497    2780  4      RECOUNT = .CCB [RAB$W_RSZ] + (IF .CCB [LUB$V_TERM_FOR] THEN SELECTONEU .CCB [RAB$W_STV0] OF
1498    2781  4      SET
1499    2782  4      [K_ESCAPE] : .CCB [RAB$W_STV2];
1500    2783  4      [K_R] : 2;
1501    2784  4      [OTHERWISE] : 0;
1502    2785  3      TES ELSE 0);
1503    2786  3      STATUS = 1;
1504    2787  3      END
1505    2788  2      ELSE
1506    2789  2      STATUS = 0;
1507    2790  2
1508    2791  2      RETURN .STATUS;
1509    2792  1      END;                                ! End of BASS$REC_MIN1

```

| 52 DD 00000 BASS\$REC_MIN1:: | | | | | | |
|------------------------------|-----------|-------|------|---------------------|------------------------------|------|
| 03 | 97 | AB | 01 | E0 00002 | PUSHL R2 | 2655 |
| | | | 009E | 31 00007 | BBS #1, -105(CCB), 1\$ | 2716 |
| | | | 5B | DD 0000A 1\$: | BRW 11\$ | |
| | | | 01 | FB 0000C | PUSHL CCB | 2720 |
| | 00000000G | 00 | 50 | D0 00013 | CALLS #1, SYSSGET | |
| | 00010651 | 52 | 52 | D1 00016 | MOVL R0, RMS_STATUS | |
| | | 8F | 07 | 12 0001D | CMPL RMS_STATUS, #67153 | 2722 |
| | 00000000G | 00 | 00 | FB 0001F | BN EQ 2\$ | |
| | | 07 | 52 | E8 00026 2\$: | CALLS #0, BASS\$SIGNAL_CTRLC | 2724 |
| | | | 7E | D4 00029 | BLBS RMS_STATUS, 3\$ | 2726 |
| | | | 01 | FB 0002B | CLRL -(SP) | 2728 |
| B0 | AB | 0000V | CF | C3 00030 3\$: | CALLS #1, GET_ERROR | |
| | | 28 | AB | 01 | SUBL3 #1, 40(CCB), -80(CCB) | 2746 |
| | | | 50 | 3C 00036 | MOVZWL 34(CCB), R0 | 2747 |
| | | B4 | AB | 22 BB40 9E 0003A | MOVAB @40(CCB)[R0], -76(CCB) | |
| | | | 50 | B4 AB D0 00040 | MOVL -76(CCB), R0 | 2755 |
| | | | 26 | FF A0 91 00044 | CMPB -1(R0), #38 | |
| | | | | 09 12 00048 | BNEQ 4\$ | |
| | | | B4 | AB D7 0004A | DECL -76(CCB) | 2758 |
| | | | 97 | AB 02 88 0004D | BISB2 #2, -105(CCB) | 2759 |
| | | | | 04 11 00051 | BRB 5\$ | 2755 |
| | | | 97 | AB 02 8A 00053 4\$: | BICB2 #2, -105(CCB) | 2762 |
| | | | 50 | AB D0 00057 5\$: | MOVL -72(CCB), T_CCB | 2769 |
| | | | 52 | 3C 0005B | MOVZWL 12(CCB), R2 | 2770 |
| | | | 1B | B1 0005F | CMPW R2, #27 | |
| | | | | 11 12 00062 | BNEQ 6\$ | |
| | | | OC | FE A0 02 E1 00064 | BBC #2, -2(T_CCB), 6\$ | |

| | | | | | | | | |
|--------------|--|----|----|-------|----------|--------------|--------------------|------|
| | | 51 | 22 | AB | 3C 00069 | MOVZWL | 34(CC(B), R1 | 2771 |
| | | 51 | C8 | A0 | C0 0006D | ADDL2 | -56(T_CC(B)), R1 | |
| | | 51 | | D6 | 00071 | INCL | R1 | |
| | | 02 | | 11 | 00073 | BRB | 7\$ | |
| | | 51 | | D4 | 00075 | 6\$: CLR | R1 | 2770 |
| | | 51 | | D0 | 00077 | 7\$: MOVL | R1. -56(T_CC(B) | |
| | | 04 | | E1 | 0007B | BBC | #4. -2(CC(B)), 9\$ | 2780 |
| | | 52 | | B1 | 00080 | CMPW | R2, #27 | 2782 |
| | | 06 | | 12 | 00083 | BNEQ | 8\$ | |
| | | 50 | OE | AB | 3C 00085 | MOVZWL | 14(CC(B), R0 | |
| | | 0C | | 11 | 00089 | BRB | 10\$ | |
| | | 0D | | B1 | 0008B | 8\$: CMPW | R2, #13 | 2783 |
| | | 05 | | 12 | 0008E | BNEQ | 9\$ | |
| | | 50 | | 02 | D0 00090 | MOVL | #2, R0 | |
| | | 02 | | 11 | 00093 | BRB | 10\$ | |
| | | 50 | | D4 | 00095 | 9\$: CLR | R0 | 2780 |
| 00000000' EF | | 51 | 22 | AB | 3C 00097 | 10\$: MOVZWL | 34(CC(B), R1 | |
| | | 50 | | C1 | 0009B | ADDL3 | R1, R0, RECOUNT | 2786 |
| | | 50 | | 01 | D0 000A3 | MOVL | #1, STATUS | 2716 |
| | | 02 | | 11 | 000A6 | BRB | 12\$ | 2789 |
| | | 50 | | D4 | 000A8 | 11\$: CLRL | STATUS | |
| | | 04 | | BA | 000AA | 12\$: POPR | #^M<R2> | 2792 |
| | | 05 | | 000AC | | RSB | | |

; Routine Size: 173 bytes, Routine Base: _BASS\$CODE + 03DF

; 1510 2793 1

```
: 1512      2794 1 GLOBAL ROUTINE BASS$REC_RSL9           ! Read IO_END
: 1513      2795 1 : JSB_REC9 NOVALUE ≡
: 1514      2796 1 ++
: 1515      2797 1 FUNCTIONAL DESCRIPTION:
: 1516      2798 1 BASS$REC_RSL9 is a no-op!
: 1517      2799 1
: 1518      2800 1 FORMAL PARAMETERS:
: 1519      2801 1     NONE
: 1520      2802 1
: 1521      2803 1
: 1522      2804 1
: 1523      2805 1
: 1524      2806 1
: 1525      2807 1
: 1526      2808 1
: 1527      2809 1
: 1528      2810 1 IMPLICIT INPUTS:
: 1529      2811 1     NONE
: 1530      2812 1 IMPLICIT OUTPUTS:
: 1531      2813 1 ROUTINE VALUE:
: 1532      2814 1     NONE
: 1533      2815 1 SIDE EFFECTS:
: 1534      2816 1
: 1535      2817 1
: 1536      2818 1
: 1537      2819 1
: 1538      2820 2   BEGIN
: 1539      2821 2   RETURN;
: 1540      2822 1   END;                                ! End of BASS$REC_RSL9
```

05 00000 BASS\$REC_RSL9::
RSB

: 2822

: Routine Size: 1 bytes, Routine Base: _BASS\$CODE + 048C

: 1541 2823 1

```
: 1543      2824 1 GLOBAL ROUTINE BASS$REC_MIN9          ! MAT Input IO-END
: 1544      2825 1 : JSB_REC9 NOVALUE =
: 1545      2826 1
: 1546      2827 1 ++
: 1547      2828 1 FUNCTIONAL DESCRIPTION:
: 1548      2829 1     BASS$REC_RSL9 is a no-op!
: 1549      2830 1
: 1550      2831 1 FORMAL PARAMETERS:
: 1551      2832 1     NONE
: 1552      2833 1
: 1553      2834 1 IMPLICIT INPUTS:
: 1554      2835 1     NONE
: 1555      2836 1
: 1556      2837 1 IMPLICIT OUTPUTS:
: 1557      2838 1
: 1558      2839 1 ROUTINE VALUE:
: 1559      2840 1     NONE
: 1560      2841 1
: 1561      2842 1 SIDE EFFECTS:
: 1562      2843 1     NONE
: 1563      2844 1
: 1564      2845 1
: 1565      2846 1
: 1566      2847 1
: 1567      2848 1 --
: 1568      2849 1
: 1569      2850 2     BEGIN
: 1570      2851 2     RETURN;
: 1571      2852 1     END;                                ! End of BASS$REC_MIN9
```

05 00000 BASS\$REC_MIN9::
RSB

: 2852

; Routine Size: 1 bytes, Routine Base: _BASS\$CODE + 048D
: 1572 2853 1

: 1574 2854 1 GLOBAL ROUTINE BASS\$REC_MLI1 : MAT Linput element transmitter
1575 2855 1
1576 2856 1
1577 2857 1
1578 2858 1
1579 2859 1
1580 2860 1
1581 2861 1
1582 2862 1
1583 2863 1
1584 2864 1
1585 2865 1
1586 2866 1
1587 2867 1
1588 2868 1
1589 2869 1
1590 2870 1
1591 2871 1
1592 2872 1
1593 2873 1
1594 2874 1
1595 2875 1
1596 2876 1
1597 2877 1
1598 2878 1
1599 2879 1
1600 2880 1
1601 2881 1
1602 2882 1
1603 2883 1
1604 2884 1
1605 2885 1
1606 2886 1
1607 2887 1
1608 2888 1
1609 2889 1
1610 2890 1
1611 2891 1
1612 2892 1
1613 2893 1
1614 2894 1
1615 2895 2
1616 2896 2
1617 2897 2
1618 2898 2
1619 2899 2
1620 2900 2
1621 2901 2
1622 2902 2
1623 2903 2
1624 2904 2
1625 2905 2
1626 2906 2
1627 2907 2
1628 2908 2
1629 2909 2
1630 2910 2

GLOBAL ROUTINE BASS\$REC_MLI1 : MAT Linput element transmitter

FUNCTIONAL DESCRIPTION:
BASS\$REC_MLI1 unconditionally reads one record. There is no continuation character for MAT LINPUT. Otherwise an error is signalled. Then return start and end+1 of user part of record to be processed as input.

FORMAL PARAMETERS:
NONE

IMPLICIT INPUTS:
LUB\$W_RBUF_SIZE Size of record buffer allocated in OPEN.
LUB\$A_RBUF_ADDR Address of record buffer from OPEN.
RAB\$L_RBF Pointer to buffer
RAB\$W_RSZ buffer size

IMPLICIT OUTPUTS:
RECOUNT Own storage for RECOUNT function.
LUB\$A_BUF_PTR points to first char of user part of record buffer.
LUB\$A_BUF_END points to end+1 of user part of record buffer.

ROUTINE VALUE:
NONE

SIDE EFFECTS:
Reads next record from file on this logical unit.
SIGNS any resultant RMS errors.

--

BEGIN

EXTERNAL REGISTER
CCB : REF BLOCK [, BYTE];

LOCAL
RMS STATUS,
T_CCB : REF BLOCK [, BYTE];

LITERAL
K_ESCAPE = %X'1B',
K_CR = %X'0D';

!+
! Read record into buffer using RMS and check for errors
! Signal any RMS errors directly.

```

1631      2911 2      !-
1632      2912 2
1633      2913 2      RMS_STATUS = $GET (RAB = .CCB);
1634      2914 2
1635      2915 2      IF .RMS_STATUS EQL RMSS_CONTROL.C
1636      2916 2      THEN
1637      2917 2          BAS$SIGNAL_CTRLC ();
1638      2918 2
1639      2919 2      IF NOT .RMS_STATUS
1640      2920 2      THEN
1641      2921 2          GET_ERROR (K_STOP);
1642      2922 2
1643      2923 2
1644      2924 2      + Return start-1 and end+1 address of record just read
1645      2925 2      LUBSA_BUF_PTR is set to the beginning-1 of the buffer only for BASIC
1646      2926 2      Input. This is seen as a solution to the problem of the user entering
1647      2927 2      <return> as the response to a prompt (null input record) and an empty
1648      2928 2      or depleted buffer which requires another Get.
1649      2929 2      The algorithm:
1650      2930 2          1) Does LUBSA_BUF_PTR = LUBSA_BUF_END?
1651      2931 2              T: The buffer is depleted - another Get is required.
1652      2932 2          2) Add one to LUBSA_BUF_PTR
1653      2933 2          3) Does LUBSA_BUF_PTR = LUBSA_BUF-END?
1654      2934 2              T: Return the default value.
1655      2935 2          4) Scan for the next field.
1656      2936 2
1657      2937 2
1658      2938 2      - .CCB [LUBSA_BUF_PTR] = .CCB [RAB$L_RBF] - 1;
1659      2939 2      .CCB [LUBSA_BUF-END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
1660      2940 2
1661      2941 2      + Update the cursor position if this input was terminated by an escape.
1662      2942 2      Save the cursor position if last PRINT terminator was a semi or comma.
1663      2943 2      Use BUDDY_PTR 'cuz we want to use the PRINT data base for channel 0
1664      2944 2
1665      2945 2      T_CCB = .CCB [LUBSA_BUDDY_PTR];
1666      2946 3      T_CCB [LUB$L_PRINT_POS] = (IF .CCB [RAB$W_STV0] EQL K_ESCAPE AND .T_CCB [LUB$V_FORM_CHAR] EQLU 1
1667      2947 3          THEN .CCB [RAB$W_RSZ] + .T_CCB [LUB$L_PRINT_POS] + 1
1668      2948 2          ELSE 0);
1669      2949 2
1670      2950 2      + Set RECOUNT to the number of bytes read
1671      2951 2      If the file is a terminal format file, then RECOUNT has to be
1672      2952 2      adjusted for the carriage control terminator.
1673      2953 2
1674      2954 3      RECOUNT = .CCB [RAB$W_RSZ] + (IF .CCB [LUB$V_TERM_FOR] THEN SELECTONEU .CCB [RAB$W_STV0] OF
1675      2955 3          SET
1676      2956 3              [K_ESCAPE] : .CCB [RAB$W_STV2];
1677      2957 3              [K_CR] : 2;
1678      2958 3              [OTHERWISE] : 0;
1679      2959 2              TES ELSE 0);
1680      2960 2      RETURN 1
1681      2961 1      END;
                                         ! End of BASS$REC_MLI1

```

| | | | | | |
|--|--|--|--|------------------------------|------|
| | | | | PUSHL R2 | 2854 |
| | | | | PUSHL CCB | 2913 |
| | | | | CALLS #1, SYS\$GET | |
| | | | | MOVL R0, RMS STATUS | |
| | | | | CMPBL RM\$ STATUS, #67153 | |
| | | | | BNEQ 1\$ | 2915 |
| | | | | CALLS #0, BASS\$SIGNAL_CTRLC | 2917 |
| | | | | BLBS RM\$ STATUS, 2\$ | 2919 |
| | | | | CLRL -(SP) | 2921 |
| | | | | CALLS #1, GET_ERROR | |
| | | | | SUBL3 #1, 40(CCB), -80(CCB) | 2938 |
| | | | | MOVZWL 34(CCB), R0 | 2939 |
| | | | | MOVAB @40(CCB)[R0], -76(CCB) | |
| | | | | MOVL -72(CCB), T_CCB | 2945 |
| | | | | MOVZWL 12(CCB), R2 | 2946 |
| | | | | CMPW R2, #27 | |
| | | | | BNEQ 3\$ | |
| | | | | BBC #2, -2(T_CCB), 3\$ | |
| | | | | MOVZWL 34(CCB), R0 | |
| | | | | ADDL2 -56(T_CCB), R0 | 2947 |
| | | | | INCL R0 | |
| | | | | BRB 4\$ | |
| | | | | CLRL R0 | 2946 |
| | | | | MOVL R0, -56(T_CCB) | |
| | | | | BBC #4, -2(CCB), 6\$ | 2954 |
| | | | | CMPW R2, #27 | 2956 |
| | | | | BNEQ 5\$ | |
| | | | | MOVZWL 14(CCB), R0 | |
| | | | | BRB 7\$ | |
| | | | | CMPW R2, #13 | 2957 |
| | | | | BNEQ 6\$ | |
| | | | | MOVL #2, R0 | |
| | | | | BRB 7\$ | |
| | | | | CLRL R0 | 2954 |
| | | | | MOVZWL 34(CCB), R1 | |
| | | | | ADDL3 R1, R0, RECOUNT | 2960 |
| | | | | MOVL #1, R0 | 2961 |
| | | | | POPR #^M<R2> | |
| | | | | RSB | |

: Routine Size: 138 bytes. Routine Base: _BASS\$CODE + 048E

: 1682 2962 1

```
1684    2963 1 GLOBAL ROUTINE BASS$REC_WSL0           ! Write list-directed
1685    2964 1 : JSB_REC0 NOVALUE =
1686    2965 1
1687    2966 1 ++
1688    2967 1 FUNCTIONAL DESCRIPTION:
1689    2968 1
1690    2969 1 BASS$REC_WSL0 prepares a record for list-directed output.
1691    2970 1 Then return start and end+1 of user
1692    2971 1 part of record to be processed.
1693    2972 1
1694    2973 1 FORMAL PARAMETERS:
1695    2974 1
1696    2975 1     NONE
1697    2976 1
1698    2977 1 IMPLICIT INPUTS:
1699    2978 1
1700   2979 1     LUB$W_RBUF_SIZE      Size (bytes) allocated for record buffer at OPEN.
1701   2980 1     LUB$A_RBUF_ADR      Address of record buffer allocated at OPEN
1702   2981 1     LUB$V_FIXED        1 if fixed-length records
1703   2982 1     LUB$V_FORM_CHAR    Indicates that the last element transmitter ended
1704   2983 1     in a comma or semicolon format char.
1705   2984 1     LUB$V_FORCEABLE    Indicates a forcible device
1706   2985 1     LUB$V_CCO          Cancel control 0
1707   2986 1
1708   2987 1 IMPLICIT OUTPUTS:
1709   2988 1
1710   2989 1     LUB$B_BAS_VFC1      'Pre' carriage control
1711   2990 1     LUB$B_BAS_VFC2      'Post' carriage control
1712   2991 1     LUB$A_BUF_PTR       pointer to next byte of buffer
1713   2992 1     LUB$A_BUF_END      pointer to byte following the buffer
1714   2993 1     RAB$V_CCO          Cancel control 0
1715   2994 1
1716   2995 1 ROUTINE VALUE:
1717   2996 1
1718   2997 1     NONE
1719   2998 1
1720   2999 1 SIDE EFFECTS:
1721   3000 1
1722   3001 1 --
1723   3002 1
1724   3003 2 BEGIN
1725   3004 2
1726   3005 2 EXTERNAL REGISTER
1727   3006 2     CCB : REF BLOCK [, BYTE];
1728   3007 2
1729   3008 2
1730   3009 2 + Copy the current status of the cancel-control-0 bit in the LUB
1731   3010 2 (possibly set by RCTRLO) into the RAB, and clear it from the
1732   3011 2 LUB. The net effect of this is that if the bit is set in the
1733   3012 2 LUB, then the CANCTRL0 modifier will be applied to this write
1734   3013 2 operation only.
1735   3014 2 -
1736   3015 2
1737   3016 2     CCB [RAB$V_CCO] = CCB [LUB$V_CCO];
1738   3017 2     CCB [LUB$V_CCO] = 0;
1739   3018 2
1740   3019 2     !+
```

```

1741      3020  2 | If there is a format character pending and this is not a forcible
1742      3021  2 | device, then don't change the buffer pointers. The PUT will be done when
1743      3022  2 | there is no format character pending.
1744      3023  2 |
1745      3024  2 |
1746      3025  2 | IF .CCB [LUB$V_FORM_CHAR] AND NOT .CCB [LUB$V_FORCIBLE] THEN RETURN;
1747      3026  2 |
1748      3027  2 |
1749      3028  2 | If the last statement did not end with a format character,
1750      3029  2 | then put a line feed into the 'pre' carriage control
1751      3030  2 | Unconditionally set the 'post' carriage control to null
1752      3031  2 |
1753      3032  2 |
1754      3033  2 | CCB [LUB$B_BAS_VFC1] = (IF .CCB [LUB$V_FORM_CHAR] THEN BASSK_NULL ELSE BASSK_LF);
1755      3034  2 | CCB [LUB$B_BAS_VFC2] = BASSK_NULL;
1756      3035  2 |
1757      3036  2 | If the buffer is dirty, then this is recursive I/O and we want to
1758      3037  2 | concatenate this record. So leave the buffer pointers alone. Otherwise
1759      3038  2 | return the buffer pointers initialized for another statement
1760      3039  2 |
1761      3040  2 |
1762      3041  2 |
1763      3042  2 | IF NOT .CCB [LUB$V_OUTBUF_DR]
1764      3043  2 | THEN
1765      3044  3 | BEGIN
1766      3045  3 |   CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADDR];
1767      3046  3 |   CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADDR] + .CCB [LUB$W_RBUF_SIZE];
1768      3047  2 | END;
1769      3048  2 |
1770      3049  2 | RETURN;
1771      3050  1 | END;

```

! END OF BASS\$REC_WSL0

| | | | | | | | | |
|----|----|----|----|----|----|---------------------------------------------|--|------|
| | 50 | A0 | AB | 01 | 02 | EF 00000 BASS\$REC_WSL0:: | | |
| 07 | AB | 01 | | 07 | 50 | F0 00006 EXTZV #2, #1, -96(CCB), R0 | | 3016 |
| | | A0 | AB | | 04 | 8A 0000C INSV R0, #7, #1, 7(CCB) | | 3017 |
| | | FE | AB | | 02 | E1 00010 BICB2 #4, -96(CCB) | | 3025 |
| | | 24 | FE | AB | 06 | E1 00015 BBC #2, -2(CCB), 1\$ | | |
| | | 04 | FE | AB | 02 | E1 0001A BBC #6, -2(CCB), 3\$ | | 3033 |
| | | | | | 50 | D4 0001F CLRL #2, -2(CCB), 1\$ | | |
| | | | | | 03 | 11 00021 BRB 2\$ | | |
| | | | | | 01 | D0 00023 1\$: MOVL #1, R0 | | |
| | | | | | 50 | 9B 00026 2\$: MOVZBW R0, -38(CCB) | | 3042 |
| | | OF | DA | AB | 03 | E0 0002A BBS #3, -2(CCB), 3\$ | | 3045 |
| | | | FE | AB | AB | D0 0002F MOVL -20(CCB), -80(CCB) | | 3046 |
| | | | B0 | AB | EC | 3C 00034 MOVZWL -46(CCB), R0 | | |
| | | | | | 50 | BB40 9E 00038 MOVAB a-20(CCB)[R0], -76(CCB) | | 3050 |
| | | | | | D2 | AB 05 0003E 3\$: RSB | | |

; Routine Size: 63 bytes, Routine Base: _BASS\$CODE + 0518

; 1772 3051 1

```
: 1774      3052 1 GLOBAL ROUTINE BASS$REC_WSL1 (           ! Write list-directed
: 1775      3053 1   CARRIAGE_CTRL) : JSB_REC_WSL1 NOVALUE =           ! Called from BASS$DO_WRITE
: 1776      3054 1
: 1777      3055 1 ++
: 1778      3056 1   FUNCTIONAL DESCRIPTION:
: 1779      3057 1
: 1780      3058 1   Write one list-directed record and initialize for the next
: 1781      3059 1   BASS$REC_WSL1 writes one output buffer and then
: 1782      3060 1   initializes the output buffer and returns start and end+1 of user
: 1783      3061 1   part of record buffer to be filled by caller.
: 1784      3062 1   If this routine is called because the buffer overflowed then the 'post'
: 1785      3063 1   carriage control should be null. If this routine is called because the
: 1786      3064 1   margin overflowed, then the 'post' carriage control should be 'CR'.
: 1787      3065 1
: 1788      3066 1   FORMAL PARAMETERS:
: 1789      3067 1
: 1790      3068 1   CARRIAGE_CTRL.rlu.v    carriage control for the record
: 1791      3069 1
: 1792      3070 1   IMPLICIT INPUTS:
: 1793      3071 1
: 1794      3072 1   LUB$W_RBUF_SIZE          Size (bytes) allocated for record buffer at OPEN.
: 1795      3073 1   LUBSA_RBUF_ADR          Address of record buffer allocated at OPEN
: 1796      3074 1   LUBSA_BUF_PTR            Pointer to next byte in user buffer.
: 1797      3075 1   RABSL_RBF                Pointer to user buffer
: 1798      3076 1
: 1799      3077 1   IMPLICIT OUTPUTS:
: 1800      3078 1
: 1801      3079 1   LUB$B_BAS_VFC1          'Pre' carriage control
: 1802      3080 1   LUB$B_BAS_VFC2          'Post' carriage control
: 1803      3081 1   LUBSA_BUF_PTR            Pointer to start of user part of record buffer
: 1804      3082 1   LUBSA_BUF_END             Pointer to end+1 of user part of record buffer
: 1805      3083 1   LUB$V_OUTBUF_DR           Indicates valid data in the output buffer
: 1806      3084 1   RABSW_RSZ                Size of user buffer
: 1807      3085 1
: 1808      3086 1   ROUTINE VALUE:
: 1809      3087 1
: 1810      3088 1   NONE
: 1811      3089 1
: 1812      3090 1   SIDE EFFECTS:
: 1813      3091 1
: 1814      3092 1   Writes one RMS sequential record.
: 1815      3093 1   SIGNALS BASS$FATSYSTIO on PUT error.
: 1816      3094 1 --
: 1817      3095 1
: 1818      3096 2   BEGIN
: 1819      3097 2
: 1820      3098 2   EXTERNAL REGISTER
: 1821      3099 2     CCB = 11 : REF BLOCK [, BYTE];
: 1822      3100 2
: 1823      3101 2   LITERAL
: 1824      3102 2     K_NO_CR = 2;
: 1825      3103 2
: 1826      3104 2   LOCAL
: 1827      3105 2     RMS_STATUS;
: 1828      3106 2
: 1829      3107 2   !
: 1830      3108 2   ! Set 'post' carriage control to CR or NULL depending on whether the margin
```

```

: 1831      3109 2 | overflowed or the buffer overflowed.
: 1832      3110 2 | If this is a file, the carriage control is ignored and a record is PUT.
: 1833      3111 2 |
: 1834      3112 2 | CCB [LUB$B_BAS_VFC2] = (IF .CARRIAGE_CTRL EQL BASSK_BUF_EXC THEN BASSK_NULL ELSE BASSK_(R));
: 1835      3113 2 |
: 1836      3114 2 | +
: 1837      3115 2 | perform the record write.
: 1838      3116 2 | Set recordsize to actual length of record
: 1839      3117 2 |
: 1840      3118 2 |
: 1841      3119 2 | CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_BUF_BEG];
: 1842      3120 2 |
: 1843      3121 2 | +
: 1844      3122 2 | Output buffer to RMS and check for errors
: 1845      3123 2 | If errors, SIGNAL_STO
: 1846      3124 2 |
: 1847      3125 2 |
: 1848      3126 2 | CCB [RAB$L_RBF] = .CCB [LUB$A_BUF_BEG];
: 1849      3127 2 | CCB [LUB$V_OUTBUF_DR] = 0;
: 1850      3128 2 |
: 1851      3129 2 | RMS_STATUS = $PUT (RAB = .CCB);
: 1852      3130 2 |
: 1853      3131 2 | IF .RMS_STATUS EQL RMSS_CONTROLC
: 1854      3132 2 | THEN
: 1855      3133 2 |     BASS$SIGNAL_CTRLC ();
: 1856      3134 2 |
: 1857      3135 2 | IF NOT .RMS_STATUS
: 1858      3136 2 | THEN
: 1859      3137 2 |     PUT_ERROR (K_STOP);
: 1860      3138 2 |
: 1861      3139 2 | +
: 1862      3140 2 | Set the 'pre' carriage control to LF if CARRIAGE_CTRL warrants it.
: 1863      3141 2 | Set the 'post' carriage control to null.
: 1864      3142 2 |
: 1865      3143 2 |
: 1866      3144 2 | CCB [LUB$B_BAS_VFC1] = (IF .CARRIAGE_CTRL EQL BASSK_BUF_EXC THEN BASSK_NULL ELSE BASSK_LF);
: 1867      3145 2 | CCB [LUB$B_BAS_VFC2] = BASSK_NULL;
: 1868      3146 2 |
: 1869      3147 2 | +
: 1870      3148 2 | Initialize record buffer for another list-directed write
: 1871      3149 2 | return record buffer pointers to caller
: 1872      3150 2 |
: 1873      3151 2 |
: 1874      3152 2 | CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
: 1875      3153 2 | CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
: 1876      3154 2 | RETURN;
: 1877      3155 1 | END;                                ! End of routine - BASS$UDF_WSL1

```

OC BB 00000 BASS\$REC_WSL1::
 08 53 D4 00002 PUSHR #^M<R2,R3>
 50 D1 00004 CLRL R3
 06 12 00007 CMPL CARRIAGE_CTRL, #8
 BNEQ 1\$

3052
3112

| | | | | | | | |
|----|----|----|-----------|------------------|------|--------|-----------------------------|
| | | | 53 | D6 00009 | INCL | R3 | |
| | | | 50 | D4 0000B | CLRL | R0 | |
| | | | 04 | 11 0000D | BRB | 2\$ | |
| 22 | AB | DB | 8F | 9A 0000F | 1\$: | MOVZBL | #141, R0 |
| | | AB | 50 | 90 00013 | 2\$: | MOVB | R0 -37(CCB) |
| | | BO | AB | AB A3 00017 | | SUBW3 | -68(CCB), -80(CCB), 34(CCB) |
| | | 28 | AB | AB D0 0001E | | MOVL | -68(CCB), 40(CCB) |
| | | FE | AB | 08 8A 00023 | | BICB2 | #8, -2(CCB) |
| | | | 5B | DD 00027 | | PUSHL | CCB |
| | | | 01 | FB 00029 | | CALLS | #1, SYSSPUT |
| | | | 52 | 50 D0 00030 | | MOVL | R0, RMS STATUS |
| | | | 00000000G | 52 D1 00033 | | CMPL | RMS_STATUS, #67153 |
| | | | 00010651 | 07 12 0003A | | BNEQ | 3\$ |
| | | | 00000000G | 00 FB 0003C | | CALLS | #0, BASS\$SIGNAL_CTRLC |
| | | | 07 | 52 E8 00043 | 3\$: | BLBS | RMS_STATUS, 4\$ |
| | | | 00000V | 7E D4 00046 | | CLRL | -(SP) |
| | | | CF | 01 FB 00048 | | CALLS | #1, PUT_ERROR |
| | | | 04 | 53 E9 0004D | 4\$: | BLBC | R3, 5\$ |
| | | | 50 | 50 D4 00050 | | CLRL | R0 |
| | | | DA | 03 11 00052 | | BRB | 6\$ |
| | | | AB | 01 D0 00054 | 5\$: | MOVL | #1, R0 |
| | | | BO | 50 9B 00057 | 6\$: | MOVZBW | R0 -38(CCB) |
| | | | 50 | EC AB D0 0005B | | MOVL | -20(CCB), -80(CCB) |
| | | | B4 | D2 AB 3C 00060 | | MOVZWL | -46(CCB), R0 |
| | | | AB | EC BB40 9E 00064 | | MOVAB | 0-20(CCB)[R0], -76(CCB) |
| | | | | OC BA 0006A | | POPR | #^M<R2,R3> |
| | | | | 05 0006C | | RSB | |

: Routine Size: 109 bytes, Routine Base: _BASS\$CODE + 0557

: 1878 3156 1

```
; 1880      3157 1 GLOBAL ROUTINE BASS$REC_RMFO           ! Initialize read memory formatted
; 1881      3158 1 : JSB_REC0 NOVALUE =
; 1882
; 1883
; 1884      3160 1 ++
; 1885      3161 1 FUNCTIONAL DESCRIPTION:
; 1886      3162 1
; 1887      3163 1 Pick up pointer to last major frame from ISB and initialize BUF_BEG,
; 1888      3164 1 BUF_PTR, and BUF_END to the values for the data area found in the
; 1889      3165 1 frame.
; 1890      3166 1
; 1891      3167 1 FORMAL PARAMETERS:
; 1892      3168 1     NONE
; 1893      3169 1
; 1894      3170 1 IMPLICIT INPUTS:
; 1895      3171 1
; 1896      3172 1     ISBSA_MAJ_F_PTR          pointer to last Basic major frame
; 1897      3173 1
; 1898      3174 1 IMPLICIT OUTPUTS:
; 1899      3175 1
; 1900      3176 1 ROUTINE VALUE:
; 1901      3177 1     NONE
; 1902      3178 1
; 1903      3179 1
; 1904      3180 1
; 1905      3181 1 SIDE EFFECTS:
; 1906      3182 1     NONE
; 1907      3183 1
; 1908      3184 1
; 1909      3185 1
; 1910      3186 1
; 1911      3187 2
; 1912      3188 2
; 1913      3189 2 BEGIN
; 1914      3190 2
; 1915      3191 2 EXTERNAL REGISTER
; 1916      3192 2     CCB = K_CCB_REG : REF BLOCK [, BYTE];
; 1917      3193 2
; 1918      3194 2 LOCAL
; 1919      3195 2     BMF : REF BLOCK [0, BYTE] FIELD (BSF$MAJOR_FRAME);
; 1920      3196 2
; 1921      3197 2     !+ Reach back into the last major frame by picking up the value of R11 stored
; 1922      3198 2     in the ISB. Initialize BUF_PTR, BUF_END, BUF_BEG so that this will look
; 1923      3199 2     like a vanilla INPUT.
; 1924      3200 2
; 1925      3201 2     !-
; 1926      3202 2     BMF = .CCB [ISBSA_MAJ_F_PTR];
; 1927      3203 2
; 1928      3204 2     !+ If this cell is zero, then there was no DATA statement and an error should be
; 1929      3205 2     signalled.
; 1930      3206 2
; 1931      3207 2     !-
; 1932      3208 2     IF .BMF [BSF$A_CUR_DTA] EQLA 0 THEN BASS$STOP_IO (BASS$K_OUTOF_DAT);
; 1933      3209 2
; 1934      3210 2     CCB [LUB$A_BUF_BEG] = .BMF [BSF$A_CUR_DTA];
; 1935      3211 2     CCB [LUB$A_BUF_END] = .BMF [BSF$A_END_DTA];
; 1936      3212 2
; 1937      3213 2     !+ Subtract one from CUR_DATA for INPUT element transmitter compatibility.
```

: 1937 3214 2 :-
: 1938 3215 2
: 1939 3216 2 CCB [LUB\$A_BUF_PTR] = .BMF [BSF\$A_CUR_DTA] - 1;
: 1940 3217 2 RETURN;
: 1941 3218 1 END;

OC BB 00000 BASS\$REC RMFO::
52 FF48 CB D0 00002 PUSHR #^M<R2,R3> : 3157
53 0087 C2 D0 00007 MOVL -184(CC(B)), BMF : 3201
00000000G 7E 006 0B 12 0000C MOVL 135(BMF), R3 : 3207
BC AB 00 8F 9A 0000E BNEQ 1\$
B4 AB 008B 01 FB 00012 MOVZBL #BASS\$K OUTOF DAT, -(SP)
B0 AB FF 53 D0 00019 1\$: CALLS #1, BASS\$STOP_IO : 3209
OC BA 00028 05 0002A MOVL R3, -68(CC(B)) : 3210
RSB POPR -1(R3), -80(CC(B)) : 3216
MOVAB #^M<R2,R3> : 3218

: Routine Size: 43 bytes, Routine Base: _BASS\$CODE + 05C4

: 1942 3219 1

```
; 1944      3220 1 GLOBAL ROUTINE BASS$REC_MRE1           ! Mat Read element transmitter
; 1945      3221 1 : JSB_REC1 =
; 1946      3222 1
; 1947      3223 1 ++
; 1948      3224 1 FUNCTIONAL DESCRIPTION:
; 1949      3225 1
; 1950      3226 1 Since MAT READ just takes as much input data as it can get, it will just
; 1951      3227 1 return a failure here because there is no more data.
; 1952      3228 1
; 1953      3229 1 FORMAL PARAMETERS:
; 1954      3230 1
; 1955      3231 1     NONE
; 1956      3232 1
; 1957      3233 1 IMPLICIT INPUTS:
; 1958      3234 1
; 1959      3235 1     NONE
; 1960      3236 1
; 1961      3237 1 IMPLICIT OUTPUTS:
; 1962      3238 1
; 1963      3239 1     NONE
; 1964      3240 1
; 1965      3241 1 ROUTINE VALUE:
; 1966      3242 1
; 1967      3243 1     Returns failure - out of data.
; 1968      3244 1
; 1969      3245 1 SIDE EFFECTS:
; 1970      3246 1
; 1971      3247 1 As a result of the failure being returned, the MAT READ will stop
; 1972      3248 1 filling the matrix.
; 1973      3249 1
; 1974      3250 1 --
; 1975      3251 1
; 1976      3252 2 BEGIN
; 1977      3253 2 RETURN 0
; 1978      3254 1 END;                                ! end of BASS$REC_MRE1
```

50 D4 00000 BASS\$REC_MRE1::
05 00002 CLR1 R0
 RSB

; Routine Size: 3 bytes, Routine Base: _BASS\$CODE + 05EF

; 1979 3255 1

: 3253
: 3254

```

: 1981      3256 1 GLOBAL ROUTINE BASS$REC_RMF1           ! Read element transmitter
: 1982      3257 1 : JSB_REC1 NOVALUE =
: 1983      3258 1
: 1984      3259 1 ++
: 1985      3260 1 FUNCTIONAL DESCRIPTION:
: 1986      3261 1
: 1987      3262 1 BASS$REC_RMF1 should not be called in normal processing and will signal
: 1988      3263 1 an error (BASS$K_OUTOF_DAT) if it is called.
: 1989      3264 1
: 1990      3265 1 FORMAL PARAMETERS:
: 1991      3266 1
: 1992      3267 1     NONE
: 1993      3268 1
: 1994      3269 1 IMPLICIT INPUTS:
: 1995      3270 1     NONE
: 1996      3271 1
: 1997      3272 1 IMPLICIT OUTPUTS:
: 1998      3273 1
: 1999      3274 1 ROUTINE VALUE:
: 2000      3275 1     NONE
: 2001      3276 1
: 2002      3277 1
: 2003      3278 1
: 2004      3279 1 SIDE EFFECTS:
: 2005      3280 1
: 2006      3281 1     Signal - BASS$K_OUTOF_DAT - Out of Data
: 2007      3282 1
: 2008      3283 1 --
: 2009      3284 1
: 2010      3285 2 BEGIN
: 2011      3286 2     BASS$SIGNAL (BASS$K_OUTOF_DAT);
: 2012      3287 2     RETURN;
: 2013      3288 1     END;                                ! end of BASS$REC_RMF1

```

| | | | | | | | |
|-----------|----|----|-----|----|-------|------------------------|--------|
| 00000000G | 00 | 7E | 00G | 8F | 9A | 00000 BASS\$REC_RMF1:: | : 3286 |
| | | | | 01 | FB | 00004 | |
| | | | | 05 | 0000B | CALLS #1, BASS\$SIGNAL | : 3288 |
| | | | | | | RSB | |

; Routine Size: 12 bytes, Routine Base: _BASS\$CODE + 05F2

; 2014 3289 1

```

: 2016    3290 1 GLOBAL ROUTINE BASS$REC_RMF9           ! Read IO_END
: 2017    3291 1 : JSB_REC9 NOVALUE =
: 2018    3292 1
: 2019    3293 1 ++
: 2020    3294 1 FUNCTIONAL DESCRIPTION:
: 2021    3295 1
: 2022    3296 1     Update the current data pointer in the last Basic major frame
: 2023    3297 1
: 2024    3298 1 FORMAL PARAMETERS:
: 2025    3299 1
: 2026    3300 1     NONE
: 2027    3301 1
: 2028    3302 1 IMPLICIT INPUTS:
: 2029    3303 1
: 2030    3304 1     NONE
: 2031    3305 1
: 2032    3306 1 IMPLICIT OUTPUTS:
: 2033    3307 1
: 2034    3308 1 ROUTINE VALUE:
: 2035    3309 1
: 2036    3310 1     NONE
: 2037    3311 1
: 2038    3312 1 SIDE EFFECTS:
: 2039    3313 1
: 2040    3314 1 --
: 2041    3315 1
: 2042    3316 2 BEGIN
: 2043    3317 2
: 2044    3318 2 EXTERNAL REGISTER
: 2045    3319 2     CCB = K_CCB_REG : REF_BLOCK [0, BYTE];
: 2046    3320 2
: 2047    3321 2 LOCAL
: 2048    3322 2     BMF : REF_BLOCK [0, BYTE] FIELD (BSF$MAJOR_FRAME);
: 2049    3323 2
: 2050    3324 2
: 2051    3325 2
: 2052    3326 2     Set the current data pointer in the frame to BUF_PTR + 1.
: 2053    3327 2     The one is added because Input initialize will subtract one from BUF_PTR.
: 2054    3328 2     This whole matter is explained in IO_BEG.
: 2055    3329 2
: 2056    3330 2
: 2057    3331 2     BMF = .CCB [ISB$A_MAJ_F_PTR];
: 2058    3332 2     BMF [BSF$A_CUR_DTA] = -.CCB [LUB$A_BUF_PTR] + 1;
: 2059    3333 2     RETURN;
: 2060    3334 1     END;                                ! End of routine BASS$REC_RMF9

```

| | | | | | | | |
|------|----|----|------|----|---------------------------|-------|-------------------------|
| | | 50 | FF48 | CB | DO 00000 BASS\$REC RMF9:: | | |
| 0087 | C0 | B0 | AB | 01 | C1 00005 | MOVL | -184(CC(B), BMF |
| | | | | 05 | 0000C | ADDL3 | #1, -80(CC(B), 135(BMF) |
| | | | | | | RSB | |

; Routine Size: 13 bytes, Routine Base: _BASS\$CODE + 05FE

: 3330
: 3331
: 3333

BASS\$REC_PROC
1-095

: 2060 3334 1

J 8
16-Sep-1984 01:01:12
14-Sep-1984 11:56:35

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASREC[PRO.B32;1]

Page 56
(23)

```

2062      3335 1 GLOBAL ROUTINE BASS$REC_GSE (
2063          3336 1   FOREIGN_BUFFER,
2064          3337 1   LOCK_FLAGS
2065          3338 1 ) : JSB_DO_READ NOVALUE =
2066          3339 1
2067          3340 1 ++
2068          3341 1   FUNCTIONAL DESCRIPTION:
2069          3342 1
2070          3343 1   Read one record. Update RECOUNT if successful.
2071          3344 1   If a foreign buffer is specified, then change RABSL_RBF to point to the
2072          3345 1   "foreign buffer". Otherwise, signal a fatal error.
2073          3346 1
2074          3347 1   FORMAL PARAMETERS:
2075          3348 1
2076          3349 1   FOREIGN_BUFFER.rl.v           points to CB of foreign buffer or 0
2077          3350 1   LOCK_FLAGS.rlu.v           bits to set in RAB ROP for manual
2078          3351 1                           record locking (0 if none)
2079          3352 1
2080          3353 1
2081          3354 1   RABSW_USZ                  User buffer size of foreign buffer
2082          3355 1   RABSL_UBF                  Pointer to user buffer for foreign buffer
2083          3356 1   LUBSL_WAIT_TIME           Wait time for next input
2084          3357 1   WAIT                      The module level OWN WAIT
2085          3358 1
2086          3359 1
2087          3360 1
2088          3361 1   RABSB_RAC                  Record access field
2089          3362 1   RECOUNT                   Own storage for RECOUNT function.
2090          3363 1   RABSL_RBF                  Record pointer in RAB.
2091          3364 1   RABSW_RSZ                  Number of bytes read (stored in RECOUNT)
2092          3365 1
2093          3366 1
2094          3367 1
2095          3368 1   NONE
2096          3369 1
2097          3370 1
2098          3371 1
2099          3372 1
2100          3373 1   RABSW_USZ and RABSW_UBF are altered while this routine is running,
2101          3374 1   but are restored before exit.
2102          3375 1   Reads next record from file on this logical unit.
2103          3376 1   SIGNALS any RMS errors
2104          3377 1
2105          3378 2
2106          3379 2
2107          3380 2   EXTERNAL REGISTER
2108          3381 2   CCB : REF BLOCK [, BYTE];
2109          3382 2
2110          3383 2
2111          3384 2
2112          3385 2
2113          3386 2
2114          3387 2
2115          3388 2
2116          3389 2
2117          3390 2
2118          3391 2   LOCAL
2119          3392 2   RMS_STATUS.
2120          3393 2   SAVE_USZ.
2121          3394 2   ACTUAL_RSZ.
2122          3395 2   WAIT_TIME;
2123          3396 2
2124          3397 2
2125          3398 2
2126          3399 2
2127          3400 2
2128          3401 2
2129          3402 2
2130          3403 2
2131          3404 2
2132          3405 2
2133          3406 2
2134          3407 2
2135          3408 2
2136          3409 2
2137          3410 2
2138          3411 2
2139          3412 2
2140          3413 2
2141          3414 2
2142          3415 2
2143          3416 2
2144          3417 2
2145          3418 2
2146          3419 2
2147          3420 2
2148          3421 2
2149          3422 2
2150          3423 2
2151          3424 2
2152          3425 2
2153          3426 2
2154          3427 2
2155          3428 2
2156          3429 2
2157          3430 2
2158          3431 2
2159          3432 2
2160          3433 2
2161          3434 2
2162          3435 2
2163          3436 2
2164          3437 2
2165          3438 2
2166          3439 2
2167          3440 2
2168          3441 2
2169          3442 2
2170          3443 2
2171          3444 2
2172          3445 2
2173          3446 2
2174          3447 2
2175          3448 2
2176          3449 2
2177          3450 2
2178          3451 2
2179          3452 2
2180          3453 2
2181          3454 2
2182          3455 2
2183          3456 2
2184          3457 2
2185          3458 2
2186          3459 2
2187          3460 2
2188          3461 2
2189          3462 2
2190          3463 2
2191          3464 2
2192          3465 2
2193          3466 2
2194          3467 2
2195          3468 2
2196          3469 2
2197          3470 2
2198          3471 2
2199          3472 2
2200          3473 2
2201          3474 2
2202          3475 2
2203          3476 2
2204          3477 2
2205          3478 2
2206          3479 2
2207          3480 2
2208          3481 2
2209          3482 2
2210          3483 2
2211          3484 2
2212          3485 2
2213          3486 2
2214          3487 2
2215          3488 2
2216          3489 2
2217          3490 2
2218          3491 2
2219          3492 2
2220          3493 2
2221          3494 2
2222          3495 2
2223          3496 2
2224          3497 2
2225          3498 2
2226          3499 2
2227          3500 2
2228          3501 2
2229          3502 2
2230          3503 2
2231          3504 2
2232          3505 2
2233          3506 2
2234          3507 2
2235          3508 2
2236          3509 2
2237          3510 2
2238          3511 2
2239          3512 2
2240          3513 2
2241          3514 2
2242          3515 2
2243          3516 2
2244          3517 2
2245          3518 2
2246          3519 2
2247          3520 2
2248          3521 2
2249          3522 2
2250          3523 2
2251          3524 2
2252          3525 2
2253          3526 2
2254          3527 2
2255          3528 2
2256          3529 2
2257          3530 2
2258          3531 2
2259          3532 2
2260          3533 2
2261          3534 2
2262          3535 2
2263          3536 2
2264          3537 2
2265          3538 2
2266          3539 2
2267          3540 2
2268          3541 2
2269          3542 2
2270          3543 2
2271          3544 2
2272          3545 2
2273          3546 2
2274          3547 2
2275          3548 2
2276          3549 2
2277          3550 2
2278          3551 2
2279          3552 2
2280          3553 2
2281          3554 2
2282          3555 2
2283          3556 2
2284          3557 2
2285          3558 2
2286          3559 2
2287          3560 2
2288          3561 2
2289          3562 2
2290          3563 2
2291          3564 2
2292          3565 2
2293          3566 2
2294          3567 2
2295          3568 2
2296          3569 2
2297          3570 2
2298          3571 2
2299          3572 2
2300          3573 2
2301          3574 2
2302          3575 2
2303          3576 2
2304          3577 2
2305          3578 2
2306          3579 2
2307          3580 2
2308          3581 2
2309          3582 2
2310          3583 2
2311          3584 2
2312          3585 2
2313          3586 2
2314          3587 2
2315          3588 2
2316          3589 2
2317          3590 2
2318          3591 2
2319          3592 2
2320          3593 2
2321          3594 2
2322          3595 2
2323          3596 2
2324          3597 2
2325          3598 2
2326          3599 2
2327          3600 2
2328          3601 2
2329          3602 2
2330          3603 2
2331          3604 2
2332          3605 2
2333          3606 2
2334          3607 2
2335          3608 2
2336          3609 2
2337          3610 2
2338          3611 2
2339          3612 2
2340          3613 2
2341          3614 2
2342          3615 2
2343          3616 2
2344          3617 2
2345          3618 2
2346          3619 2
2347          3620 2
2348          3621 2
2349          3622 2
2350          3623 2
2351          3624 2
2352          3625 2
2353          3626 2
2354          3627 2
2355          3628 2
2356          3629 2
2357          3630 2
2358          3631 2
2359          3632 2
2360          3633 2
2361          3634 2
2362          3635 2
2363          3636 2
2364          3637 2
2365          3638 2
2366          3639 2
2367          3640 2
2368          3641 2
2369          3642 2
2370          3643 2
2371          3644 2
2372          3645 2
2373          3646 2
2374          3647 2
2375          3648 2
2376          3649 2
2377          3650 2
2378          3651 2
2379          3652 2
2380          3653 2
2381          3654 2
2382          3655 2
2383          3656 2
2384          3657 2
2385          3658 2
2386          3659 2
2387          3660 2
2388          3661 2
2389          3662 2
2390          3663 2
2391          3664 2
2392          3665 2
2393          3666 2
2394          3667 2
2395          3668 2
2396          3669 2
2397          3670 2
2398          3671 2
2399          3672 2
2400          3673 2
2401          3674 2
2402          3675 2
2403          3676 2
2404          3677 2
2405          3678 2
2406          3679 2
2407          3680 2
2408          3681 2
2409          3682 2
2410          3683 2
2411          3684 2
2412          3685 2
2413          3686 2
2414          3687 2
2415          3688 2
2416          3689 2
2417          3690 2
2418          3691 2
2419          3692 2
2420          3693 2
2421          3694 2
2422          3695 2
2423          3696 2
2424          3697 2
2425          3698 2
2426          3699 2
2427          3700 2
2428          3701 2
2429          3702 2
2430          3703 2
2431          3704 2
2432          3705 2
2433          3706 2
2434          3707 2
2435          3708 2
2436          3709 2
2437          3710 2
2438          3711 2
2439          3712 2
2440          3713 2
2441          3714 2
2442          3715 2
2443          3716 2
2444          3717 2
2445          3718 2
2446          3719 2
2447          3720 2
2448          3721 2
2449          3722 2
2450          3723 2
2451          3724 2
2452          3725 2
2453          3726 2
2454          3727 2
2455          3728 2
2456          3729 2
2457          3730 2
2458          3731 2
2459          3732 2
2460          3733 2
2461          3734 2
2462          3735 2
2463          3736 2
2464          3737 2
2465          3738 2
2466          3739 2
2467          3740 2
2468          3741 2
2469          3742 2
2470          3743 2
2471          3744 2
2472          3745 2
2473          3746 2
2474          3747 2
2475          3748 2
2476          3749 2
2477          3750 2
2478          3751 2
2479          3752 2
2480          3753 2
2481          3754 2
2482          3755 2
2483          3756 2
2484          3757 2
2485          3758 2
2486          3759 2
2487          3760 2
2488          3761 2
2489          3762 2
2490          3763 2
2491          3764 2
2492          3765 2
2493          3766 2
2494          3767 2
2495          3768 2
2496          3769 2
2497          3770 2
2498          3771 2
2499          3772 2
2500          3773 2
2501          3774 2
2502          3775 2
2503          3776 2
2504          3777 2
2505          3778 2
2506          3779 2
2507          3780 2
2508          3781 2
2509          3782 2
2510          3783 2
2511          3784 2
2512          3785 2
2513          3786 2
2514          3787 2
2515          3788 2
2516          3789 2
2517          3790 2
2518          3791 2
2519          3792 2
2520          3793 2
2521          3794 2
2522          3795 2
2523          3796 2
2524          3797 2
2525          3798 2
2526          3799 2
2527          3800 2
2528          3801 2
2529          3802 2
2530          3803 2
2531          3804 2
2532          3805 2
2533          3806 2
2534          3807 2
2535          3808 2
2536          3809 2
2537          3810 2
2538          3811 2
2539          3812 2
2540          3813 2
2541          3814 2
2542          3815 2
2543          3816 2
2544          3817 2
2545          3818 2
2546          3819 2
2547          3820 2
2548          3821 2
2549          3822 2
2550          3823 2
2551          3824 2
2552          3825 2
2553          3826 2
2554          3827 2
2555          3828 2
2556          3829 2
2557          3830 2
2558          3831 2
2559          3832 2
2560          3833 2
2561          3834 2
2562          3835 2
2563          3836 2
2564          3837 2
2565          3838 2
2566          3839 2
2567          3840 2
2568          3841 2
2569          3842 2
2570          3843 2
2571          3844 2
2572          3845 2
2573          3846 2
2574          3847 2
2575          3848 2
2576          3849 2
2577          3850 2
2578          3851 2
2579          3852 2
2580          3853 2
2581          3854 2
2582          3855 2
2583          3856 2
2584          3857 2
2585          3858 2
2586          3859 2
2587          3860 2
2588          3861 2
2589          3862 2
2590          3863 2
2591          3864 2
2592          3865 2
2593          3866 2
2594          3867 2
2595          3868 2
2596          3869 2
2597          3870 2
2598          3871 2
2599          3872 2
2600          3873 2
2601          3874 2
2602          3875 2
2603          3876 2
2604          3877 2
2605          3878 2
2606          3879 2
2607          3880 2
2608          3881 2
2609          3882 2
2610          3883 2
2611          3884 2
2612          3885 2
2613          3886 2
2614          3887 2
2615          3888 2
2616          3889 2
2617          3890 2
2618          3891 2
2619          3892 2
2
```

```
2119      3392 2 | Save USZ in case it is modified. It is faster to always
2120      3393 2 | save and restore it, since that eliminates the test for foreign
2121      3394 2 | buffers and single-character mode at the end of this routine.
2122      3395 2 |
2123      3396 2 | - SAVE_USZ = .CCB [RAB$W_USZ];
2124      3397 2 |
2125      3398 2 | + If a timeout has been specified, store information in the RAB to tell
2126      3399 2 | RMS about it. If no timeout has been specified, clear the TMO bit
2127      3400 2 | in case there was an earlier timeout specified.
2128      3401 2 |
2129      3402 2 |
2130      3403 2 |
2131      3404 2 | + If WAIT is zero then use the LUB's wait. This is to provide upward compatibility
2132      3405 2 | . i.e. existing EXE's can run with the LUB wait value in V2.2.
2133      3406 2 |
2134      3407 2 | - WAIT_TIME = ( IF ( .WAIT EQL 0 ) THEN .CCB [ LUB$L_WAIT_TIME ] ELSE .WAIT );
2135      3408 2 |
2136      3409 2 | IF (.WAIT_TIME EQL 0)
2137      3410 2 | THEN
2138      3411 2 |   CCB [RAB$V_TMO] = 0
2139      3412 2 | ELSE
2140      3413 2 |   BEGIN
2141      3414 2 |     CCB [RAB$B_TMO] = .WAIT_TIME;
2142      3415 2 |     CCB [RAB$V_TMO] = 1;
2143      3416 2 |   END;
2144      3417 2 |
2145      3418 2 |
2146      3419 2 | + Set the Read-no-echo RMS bit based on the user's last call to
2147      3420 2 | ECHO or NOECHO.
2148      3421 2 |
2149      3422 2 | - CCB [RAB$V_RNE] = .CCB [LUB$V_NOECHO];
2150      3423 2 |
2151      3424 2 | + Set the record pointer field in the RAB to the user buffer. This is
2152      3425 2 | done on each element transmission and not just at OPEN because of RMS
2153      3426 2 | Locate mode.
2154      3427 2 |
2155      3428 2 | Fill the input buffer with Nulls. Basic semantics require null fill if
2156      3429 2 | the record read is shorter than the buffer.
2157      3430 2 | Set the record access field in the RAB to sequential. Perform the GET.
2158      3431 2 | If RMS returns a failure status, signal the error. If the GET is
2159      3432 2 | successful, then update recount.
2160      3433 2 |
2161      3434 2 |
2162      3435 2 | - IF (.FOREIGN_BUFFER NEQA 0)
2163      3436 2 | THEN
2164      3437 2 |   BEGIN
2165      3438 2 |
2166      3439 2 | + A foreign buffer was specified. Save off RAB$L_usz of the "file" channel
2167      3440 2 | and then substitute the appropriate values from the foreign channel into
2168      3441 2 | the file channel to make the record go directly into the foreign buffer.
2169      3442 2 |
2170      3443 2 | - CCB [RAB$W_USZ] = .FOREIGN_BUFFER [RAB$W_USZ];
2171      3444 2 |
2172      3445 2 |
2173      3446 2 |
2174      3447 2 | + If the user has called ONECHR, shrink the effective size of the
2175      3448 2 | buffer to one character, so he will get characters one at a time.
```

```
2176      3449 2 | The user must call ONECHR before each GET, so we clear the ONECHR
2177      3450 2 | flag here.
2178      3451 2 |
2179      3452 2 |
2180      3453 2 | IF (.CCB [LUB$V_ONECHR])
2181      3454 2 | THEN
2182      3455 2 | BEGIN
2183      3456 2 |   CCB [LUB$V_ONECHR] = 0;
2184      3457 2 |   CCB [RAB$W_USZ] = 1;
2185      3458 2 | END;
2186      3459 2 |
2187      3460 2 |   CCB [RAB$B_RAC] = RAB$C_SEQ;
2188      3461 2 |
2189      3462 2 | +
2190      3463 2 | Set bits in the RAB ROP (careful not to turn off ULK).
2191      3464 2 |
2192      3465 2 |
2193      3466 2 |   CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
2194      3467 2 |
2195      3468 2 |   RMS_STATUS = $GET (RAB = .CCB);
2196      3469 2 |
2197      3470 2 |   IF .RMS_STATUS EQL RMSS_CONTROLC
2198      3471 2 |   THEN
2199      3472 2 |     BAS$$SIGNAL_CTRLC ();
2200      3473 2 |
2201      3474 2 |   IF NOT .RMS_STATUS
2202      3475 2 |   THEN
2203      3476 2 |     BEGIN
2204      3477 2 |
2205      3478 2 |     + We cannot call GET_ERROR because we must restore UBF and USZ.
2206      3479 2 |
2207      3480 2 |
2208      3481 2 |     WHILE (.CCB [RAB$L_STS] EQL RMSS_RSA) DO
2209      3482 2 |       BEGIN
2210      3483 2 |         $WAIT (RAB = .CCB);
2211      3484 2 |         $GET (RAB = .CCB);
2212      3485 2 |       END;
2213      3486 2 |
2214      3487 2 |     END;
2215      3488 2 |
2216      3489 2 |
2217      3490 2 |     + Clear RAB ROP bits so that a subsequent I/O operation does not
2218      3491 2 |     inherit them.
2219      3492 2 |
2220      3493 2 |
2221      3494 2 |     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
2222      3495 2 |
2223      3496 2 |
2224      3497 2 |     + This actual record size may or may not change below. If the file is a
2225      3498 2 |     terminal device then it will get terminators tacked on to the record read.
2226      3499 2 |
2227      3500 2 |     ACTUAL_RSZ = .CCB [RAB$W_RSZ];
2228      3501 2 |
2229      3502 2 |
2230      3503 2 |
2231      3504 2 |     + Tack on the terminators when a terminal device file, just like INPUT LINE
2232      3505 2 |
```

```
2233      3506 2 IF .CCB[LUB$V_TERM_DEV]
2234      3507 2 THEN
2235      3508 2   BEGIN
2236      3509 2     LITERAL K_ESCAPE = XX'1B',
2237      3510 2     K_CR     = XX'0D',
2238      3511 2     K_CRLF   = XX'0A0D';
2239
2240      3512
2241      3513
2242      3514 1+ STV0 is the escape character, STV2 is the number of terminating characters.
2243      3515 1-
2244      3516 1-   SELECTONEU .CCB [RAB$W_STV0] OF
2245      3517 1-     SET
2246      3518 1-     [K_ESCAPE] :
2247      3519 1-     BEGIN
2248      3520 1+
2249      3521 1| If the length is one then escape is not at the end of the buffer and it
2250      3522 1| must be moved there, otherwise the escape sequence is at the end of the
2251      3523 1| buffer following the data.
2252      3524
2253      3525 4   IF .CCB [RAB$W_STV2] EQLU 1
2254      3526 4 THEN
2255      3527 5   BEGIN
2256      3528 5     IF .CCB [RAB$W_RSZ] GEQU .CCB [RAB$W_USZ]
2257      3529 5     THEN BAS$$STOP-IO (BAS$K_RECFLTOO);
2258      3530 5     CH$MOVE (1,UPLIT(K_ESCAPE),.CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
2259      3531 5     ACTUAL_RSZ = .ACTUAL_RSZ + 1;
2260      3532 5   END
2261      3533 4     ELSE
2262      3534 4       ACTUAL_RSZ = .ACTUAL_RSZ + .CCB [RAB$W_STV2];
2263      3535 3   END;
2264      3536 3   [K_CR] :
2265      3537 4   BEGIN
2266      3538 4     IF .CCB [RAB$W_RSZ] + 2 GTRU .CCB [RAB$W_USZ]
2267      3539 4     THEN BAS$$STOP-IO (BAS$K_RECFLTOO);
2268      3540 4     CH$MOVE (2,UPLIT(K_CRLF),.CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
2269      3541 4     ACTUAL_RSZ = .ACTUAL_RSZ + 2 ;
2270      3542 3   END;
2271      3543 3   [OTHERWISE] :
2272      3544 3   ;
2273      3545 3   TES;
2274      3546 2 END;
2275
2276      3548 1+
2277      3549 1| If there are no errors, null pad the buffer.
2278      3550 1-
2279      3551
2280      3552 2   IF (.CCB [RAB$W_USZ] GTR .ACTUAL_RSZ)AND .CCB [RAB$L_STS]
2281      3553 2 THEN
2282      3554 2   CH$FILL (XX'00',
2283      3555 2   .CCB [RAB$W_USZ] - .ACTUAL_RSZ, .CCB [RAB$L_UBF] + .ACTUAL_RSZ);
2284
2285      3557 1+
2286      3558 1| Before checking for errors, restore UBF and USZ, and set RECOUNT.
2287      3559 1-
2288      3560 2   CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
2289      3561 2   CCB [RAB$W_USZ] = .SAVE_USZ;
          3562 2   RECOUNT = .ACTUAL_RSZ;
```

```

: 2290      3563 2 1+
: 2291      3564 2 Any error remaining (which will be an error other than Record Stream
: 2292      3565 2 Active, RSA) is fatal.
: 2293      3566 2 1-
: 2294      3567 2
: 2295      3568 2 IF ( NOT .CCB [RAB$L_STS] ) THEN BASS$STOP_IO (BASS$IOERR_REC);
: 2296      3569 2
: 2297      3570 2 CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_RBF];
: 2298      3571 2 RETURN;
: 2299      3572 1 END;

```

! End of BASS\$REC_GSE

| | | | |
|----------|--------------|-------|------|
| 0000001B | 0060B | .BLKB | 1 |
| 00000A0D | 0060C P.AAA: | .LONG | 27 |
| | 00610 P.AAB: | .LONG | 2573 |

.EXTRN SY\$SWAIT

| 3C BB 00000 BASS\$REC_GSE:: | | | | | | | |
|-----------------------------|--|--|--------|----------------------------|--|--|------|
| | | | PUSHR | #^M<R2,R3,R4,R5> | | | 3335 |
| | | | SUBL2 | #16, SP | | | |
| | | | MOVL | R1, R4 | | | |
| | | | MOVAB | 32(CC8), 8(SP) | | | |
| | | | MOVZWL | @8(SP), SAVE_USZ | | | |
| | | | MOVZWL | WAIT, R1 | | | |
| | | | BNEQ | 1\$ | | | |
| | | | MOVL | -52(CC8), WAIT_TIME | | | |
| | | | MOVAB | 4(CC8), R2 | | | |
| | | | TSTL | WAIT_TIME | | | |
| | | | BNEQ | 2\$ | | | |
| | | | BICB2 | #2, 3(R2) | | | |
| | | | BRB | 3\$ | | | |
| | | | MOVB | WAIT TIME, 31(CC8) | | | |
| | | | BISB2 | #2, 3(R2) | | | |
| | | | INSV | -96(CC8), #0, #1, 3(R2) | | | |
| | | | TSTL | FOREIGN_BUFFER | | | |
| | | | BEQL | 4\$ | | | |
| | | | MOVW | 32(FOREIGN BUFFER), @8(SP) | | | |
| | | | BBC | #1, -96(CC8), 5\$ | | | |
| | | | BICB2 | #2, -96(CC8) | | | |
| | | | MOVW | #1, @8(SP) | | | |
| | | | CLRB | 30(CC8) | | | |
| | | | BISL2 | LOCK_FLAGS, (R2) | | | |
| | | | PUSHL | CCB | | | |
| | | | CALLS | #1, SY\$GET | | | |
| | | | MOVL | R0, RMS STATUS | | | |
| | | | CMPL | RM\$ STATUS, #67153 | | | |
| | | | BNEQ | 6\$ | | | |
| | | | CALLS | #0, BASS\$SIGNAL_CTRLC | | | |
| | | | BLBS | RM\$ STATUS, 8\$ | | | |
| | | | CMPL | 8(CC8), #99034 | | | |
| | | | BNEQ | 8\$ | | | |
| | | | PUSHL | CCB | | | |
| | | | CALLS | #1, SY\$SWAIT | | | |
| | | | PUSHL | CCB | | | |
| | | | CALLS | #1, SY\$GET | | | |

; Routine Size: 338 bytes, Routine Base: _BASS\$CODE + 0614

: 2300 3573 1

```

: 2302      3574 1 GLOBAL ROUTINE BASS$REC_GIN (
: 2303          3575 1           KEY_NO, REL_OP, KEY, FOREIGN_BUFFER, LOCK_FLAGS) : JSB_REC_IND1 NOVALUE =
: 2304          3576 1
: 2305          3577 1 ++
: 2306          3578 1 FUNCTIONAL DESCRIPTION:
: 2307          3579 1
: 2308          3580 1     Read one record. Update RECOUNT if successful.
: 2309          3581 1     If a foreign buffer is specified, then change RABSL_RBF to point to the
: 2310          3582 1     "foreign buffer". Otherwise, signal a fatal error.
: 2311          3583 1
: 2312          3584 1 FORMAL PARAMETERS:
: 2313          3585 1
: 2314          3586 1     KEY_NO.rl.v           key of reference number
: 2315          3587 1     REL_OP.rl.v          relative relationship of the key
: 2316          3588 1     KEY.rt.dx            key to search for
: 2317          3589 1     FOREIGN BUFFER.rl.v   points to CB of foreign buffer or 0
: 2318          3590 1     LOCK_FLAGS.rlu.v    bits to set in RAB ROP to control manual record
: 2319          3591 1           locking (0 if none)
: 2320          3592 1
: 2321          3593 1 IMPLICIT INPUTS:
: 2322          3594 1
: 2323          3595 1     RABSW_USZ             User buffer size of foreign buffer
: 2324          3596 1     RABSL_UBF              Pointer to user buffer for foreign buffer
: 2325          3597 1
: 2326          3598 1 IMPLICIT OUTPUTS:
: 2327          3599 1
: 2328          3600 1     RABSB_RAC              Record access field
: 2329          3601 1     RECOUNT                Own storage for RECOUNT function.
: 2330          3602 1     RABSL_RBF              Record pointer in RAB.
: 2331          3603 1     RABSW_USZ              User buffer size for "file" buffer
: 2332          3604 1     RABSL_UBF              Pointer to user buffer for "file" buffer
: 2333          3605 1
: 2334          3606 1 ROUTINE VALUE:
: 2335          3607 1
: 2336          3608 1     NONE
: 2337          3609 1
: 2338          3610 1 SIDE EFFECTS:
: 2339          3611 1
: 2340          3612 1     Reads next record from file on this logical unit.
: 2341          3613 1     SIGNALS any RMS errors
: 2342          3614 1 --
: 2343          3615 1
: 2344          3616 2 BEGIN
: 2345          3617 2
: 2346          3618 2 EXTERNAL REGISTER
: 2347          3619 2     CCB : REF_BLOCK [, BYTE];
: 2348          3620 2
: 2349          3621 2 MAP
: 2350          3622 2     KEY : REF_BLOCK [8, BYTE],           ! descriptor of the key
: 2351          3623 2     FOREIGN_BUFFER : REF_BLOCK [, BYTE];
: 2352          3624 2
: 2353          3625 2 LITERAL
: 2354          3626 2     K_EQUAL = 0,                      ! keys should be equal
: 2355          3627 2     K_GREATER_EQUAL = 1,           ! key should be greater than or equal
: 2356          3628 2     K_GREATER_THAN = 2;           ! key should be greater than
: 2357          3629 2
: 2358          3630 2 LOCAL

```

```
: 2359      3631 2      RMS STATUS.  
: 2360      3632 2      SAVE_USZ;  
: 2361      3633 2      ! Save the USZ  
: 2362      3634 2      |+ Save USZ in case it is modified. It is faster to always  
: 2363      3635 2      save and restore it, since that eliminates the test for foreign  
: 2364      3636 2      buffers and single-character mode at the end of this routine.  
: 2365      3637 2      |-  
: 2366      3638 2      SAVE_USZ = .CCB [RABSW_USZ];  
: 2367      3639 2      |+ Set the record pointer field in the RAB to the user buffer. This is  
: 2368      3640 2      done on each element transmission and not just at OPEN because of RMS  
: 2369      3641 2      Locate mode.  
: 2370      3642 2      |+ Fill the input buffer with Nulls. Basic semantics require null fill if  
: 2371      3643 2      the record read is shorter than the buffer.  
: 2372      3644 2      |+ Set the record access field in the RAB to sequential. Perform the GET.  
: 2373      3645 2      If RMS returns a failure status, signal the error. If the GET is  
: 2374      3646 2      successful, then update recount.  
: 2375      3647 2      |-  
: 2376      3648 2      IF .FOREIGN_BUFFER NEQ 0  
: 2377      3649 2      THEN  
: 2378      3650 2      BEGIN  
: 2379      3651 2      |+ A foreign buffer was specified. Substitute the appropriate values from the foreign channel into  
: 2380      3652 2      the file channel to make the record go directly into the foreign buffer.  
: 2381      3653 2      |-  
: 2382      3654 2      CCB [RAB$L_UBF] = .FOREIGN_BUFFER [RAB$L_UBF];  
: 2383      3655 2      CCB [RAB$W_USZ] = .FOREIGN_BUFFER [RAB$W_USZ];  
: 2384      3656 2      END;  
: 2385      3657 2      |+ Set the record access field to key. Set KBF to the key. Set KSZ to the  
: 2386      3658 2      the size of the key passed. Set the key of reference to the desired key.  
: 2387      3659 2      |+ Use a case statement to toggle KGT and KGE in the ROP.  
: 2388      3660 2      |-  
: 2389      3661 2      CCB [RAB$B_RAC] = RABSC KEY;  
: 2390      3662 2      CCB [RAB$L_KBF] = .KEY [DSC$A_POINTER];  
: 2391      3663 2      CCB [RAB$B_KRF] = .KEY_NO;  
: 2392      3664 2      CCB [RAB$B_KSZ] = {IF .KEY [DSC$B_DTYPE] NEQ DSC$K_DTYPE_P  
: 2393      3665 2      THEN  
: 2394      3666 2      .KEY [DSC$W_LENGTH]  
: 2395      3667 2      ELSE (.KEY [DSC$W_LENGTH]/2 + 1));  
: 2396      3668 2      CASE .REL_OP FROM K_EQUAL TO K_GREATER_THAN OF  
: 2397      3669 2      SET  
: 2398      3670 2      |+ [K_EQUAL] :  
: 2399      3671 2      CCB [RAB$V_KGE] = CCB [RAB$V_KGT] = 0;  
: 2400      3672 2      |+ [K_GREATER_EQUAL] :  
: 2401      3673 2      BEGIN  
: 2402      3674 2      CCB [RAB$V_KGE] = 1;  
: 2403      3675 2      CCB [RAB$V_KGT] = 0;
```

```
; 2416      3688 2      END;  
; 2417      3689 2  
; 2418      3690 2      [K_GREATER_THAN] :  
; 2419      3691 3      BEGIN  
; 2420      3692 3      CCB [RAB$V_KGT] = 1;  
; 2421      3693 3      CCB [RAB$V_KGE] = 0;  
; 2422      3694 2      END;  
; 2423      3695 2      TES;  
; 2424      3696 2  
; 2425      3697 2  
; 2426      3698 2      + Set bits in RAB ROP without turning off ULK.  
; 2427      3699 2      -  
; 2428      3700 2  
; 2429      3701 2      CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;  
; 2430      3702 2  
; 2431      3703 2      RMS_STATUS = $GET (RAB = .CCB);  
; 2432      3704 2  
; 2433      3705 2      IF .RMS_STATUS EQL RMSS_CONTROLC  
; 2434      3706 2      THEN  
; 2435      3707 2      BAS$$SIGNAL_CTRLC ();  
; 2436      3708 2  
; 2437      3709 2      IF NOT .RMS_STATUS  
; 2438      3710 2      THEN  
; 2439      3711 2      BEGIN  
; 2440      3712 2      + We cannot call GET_ERROR because we must restore UBF and USZ.  
; 2441      3713 2      -  
; 2442      3714 2  
; 2443      3715 3  
; 2444      3716 3      WHILE (.CCB [RAB$L_STS] EQL RMSS_RSA) DO  
; 2445      3717 4      BEGIN  
; 2446      3718 4      SWAIT (RAB = .CCB);  
; 2447      3719 4      SGET (RAB = .CCB);  
; 2448      3720 3      END;  
; 2449      3721 3  
; 2450      3722 2      END;  
; 2451      3723 2  
; 2452      3724 2  
; 2453      3725 2      + Turn off bits in RAB ROP so that subsequent I/O operations can not  
; 2454      3726 2      inherit them.  
; 2455      3727 2  
; 2456      3728 2  
; 2457      3729 2      CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;  
; 2458      3730 2  
; 2459      3731 2  
; 2460      3732 2      + If there are no errors, null pad the buffer.  
; 2461      3733 2  
; 2462      3734 2  
; 2463      3735 2      IF (.CCB [RAB$W_USZ] GTR .CCB [RAB$W_RSZ]) AND .CCB [RAB$L_STS]  
; 2464      3736 2      THEN  
; 2465      3737 2      CHSFILL (%X'00',  
; 2466      3738 2      .CCB [RAB$W_USZ] - .CCB [RAB$W_RSZ], .CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);  
; 2467      3739 2  
; 2468      3740 2  
; 2469      3741 2      + Before checking for errors, restore UBF and USZ, and set RECOUNT.  
; 2470      3742 2  
; 2471      3743 2      CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];  
; 2472      3744 2      CCB [RAB$W_USZ] = .SAVE_USZ;
```

```

: 2473      3745 2 RECOUNT = .CCB [RAB$W_RSZ];
: 2474      3746 2
: 2475      3747 2 + Any error remaining (which will be an error other than Record Stream
: 2476      3748 2 Active, RSA) is fatal.
: 2477      3749 2 -
: 2478      3750 2
: 2479      3751 2 IF ( NOT .CCB [RAB$L_STS] ) THEN BASS$STOP_IO (BASSK_IOERR_REC);
: 2480      3752 2
: 2481      3753 2 +
: 2482      3754 2 This is frosting on the cake. LUBSA_RBUF_ADR points to the record buffer for
: 2483      3755 2 MOVE. The buffer may change due to RMS Locate Mode. Currently, Locate Mode
: 2484      3756 2 is not performed on files which UPDATE or PUT. However, in anticipation that
: 2485      3757 2 RMS may add such a capability, we point RBUF_ADR off to the buffer used by PUT.
: 2486      3758 2 -
: 2487      3759 2 CCB [LUBSA_RBUF_ADR] = .CCB [RAB$L_RBF];
: 2488      3760 2 RETURN;
: 2489      3761 1 END;

```

! End of BASS\$REC_GIN

| 3C BB 00000 BASS\$REC_GIN:: | | | | | | | | |
|-----------------------------|------|------|----|-------------------|--------|-----------------------------|--|------|
| | 7E | 20 | AB | 3C 00002 | PUSHR | #M<R2,R3,R4,R5> | | 3574 |
| | | | 53 | D5 00006 | MOVZWL | 32(CCB), SAVE USZ | | 3639 |
| | | | 0A | 13 00008 | TSTL | FOREIGN_BUFFER | | 3652 |
| 24 | AB | 24 | A3 | D0 0000A | BEQL | 1\$ | | 3659 |
| 20 | AB | 20 | A3 | B0 0000F | MOVL | 36(FOREIGN_BUFFER), 36(CCB) | | 3660 |
| 1E | AB | 01 | 90 | 00014 | MOVW | 32(FOREIGN_BUFFER), 32(CCB) | | 3669 |
| 30 | AB | 04 | A2 | D0 00018 | MOVB | #1, 30(CCB) | | 3670 |
| 35 | AB | 50 | 90 | 0001D | MOVL | 4(KEY), 48(CCB) | | 3671 |
| | | 15 | A2 | 91 00021 | MOVB | KEY NO, 53(CCB) | | 3672 |
| | | | 05 | 13 00025 | CMPB | 2(KEY), #21 | | 3672 |
| | | | 52 | 62 3C 00027 | BEQL | 2\$ | | 3674 |
| | | | | 08 11 0002A | MOVZWL | (KEY), R2 | | 3674 |
| | | | 52 | 62 3C 0002C | BRB | 3\$ | | 3676 |
| | | | 52 | 02 C6 0002F | 2\$: | MOVZWL (KEY), R2 | | 3676 |
| | | | | 52 D6 00032 | DIVL2 | #2, R2 | | |
| | | | 34 | 52 90 00034 | INCL | R2 | | |
| | | | 52 | AB 9E 00038 | MOVB | R2, 52(CCB) | | 3672 |
| 02 | 00 | 04 | 51 | CF 0003C | MOVAB | 4(CCB), R2 | | 3682 |
| 0018 | 000D | 0006 | | 00040 | CASEL | REL OP, #0, #2 | | 3678 |
| | | | | | .WORD | 5S-4S,- | | |
| | | | | | | 6S-4S,- | | |
| | | | | | | 7S-4S | | |
| | | | 02 | A2 40 8F 8A 00046 | 5\$: | BICB2 #64, 2(R2) | | 3682 |
| | | | | 10 11 0004B | BRB | 8\$ | | |
| | | | 02 | A2 40 8F 88 0004D | 6\$: | BISB2 #32, 2(R2) | | 3686 |
| | | | 02 | A2 40 8F 8A 00051 | BRB | #64, 2(R2) | | 3687 |
| | | | 02 | 09 11 00056 | BRB | 9\$ | | 3678 |
| | | | 02 | A2 40 8F 88 00058 | 7\$: | BISB2 #64, 2(R2) | | 3692 |
| | | | 02 | A2 20 8A 0005D | 8\$: | BICB2 #32, 2(R2) | | 3693 |
| | | | 62 | 54 C8 00061 | 9\$: | BISL2 LOCK_FLAGS, (R2) | | 3701 |
| | | | | 5B DD 00064 | PUSHL | CCB | | 3703 |
| 00000000G | 00 | | | 01 FB 00066 | CALLS | #1, SYSSGET | | |
| | 53 | | | 50 D0 0006D | MOVL | RO, RMS STATUS | | |
| 00010651 | 8F | | | 53 D1 00070 | CMPL | RMS_STATUS, #67153 | | 3705 |

| | | | | | | | | | | | |
|----|----|----|-----------|----|-------|-------|------------------------|---------------------|----------------------|--------------------|--|
| | | | 07 | 12 | 00077 | BNEQ | 10\$ | | | 3707 | |
| | | | 00 | FB | 00079 | CALLS | #0, | BASS\$SIGNAL_CTRLC | | | |
| | | | 53 | E8 | 00080 | 10\$: | BLBS | RM\$ STATUS 12\$ | | 3709 | |
| | | 08 | AB | D1 | 00083 | 11\$: | CMPL | 8(CC(B), #99034 | | 3716 | |
| | | | 14 | 12 | 0008B | BNEQ | 12\$ | | | | |
| | | | 58 | DD | 0008D | PUSHL | CCB | | | 3718 | |
| | | | 01 | FB | 0008F | CALLS | #1, | SYSSWAIT | | | |
| | | | 58 | DD | 00096 | PUSHL | CCB | | | 3719 | |
| | | | 01 | FB | 00098 | CALLS | #1, | SYSSGET | | | |
| | | | E2 | 11 | 0009F | BRB | 11\$ | | | 3716 | |
| | | 22 | 62 | CC | 000A1 | 12\$: | XORL2 | LOCK FLAGS, (R2) | | | |
| | | AB | 20 | AB | B1 | 000A4 | CMPW | 32(CC(B), 34(CC(B) | | 3729 | |
| | | | 19 | 08 | AB | E9 | 000AB | BLEQU | 13\$ | 3735 | |
| | | | 51 | 20 | AB | 3C | 000AF | BLBC | 8(CC(B), 13\$ | | |
| | | | 50 | 22 | AB | 3C | 000B3 | MOVZWL | 32(CC(B), R1 | | |
| | | | 51 | 50 | C2 | 000B7 | MOVZWL | 34(CC(B), R0 | | 3738 | |
| | | | 50 | 22 | AB | 3C | 000BA | SUBL2 | R0, R1 | | |
| | | | 50 | 24 | AB | C0 | 000BE | MOVZWL | 34(CC(B), R0 | | |
| 51 | 00 | | 6E | 00 | 2C | 000C2 | ADDL2 | 36(CC(B), R0 | | | |
| | | | | 60 | 000C7 | MOVC5 | #0, (SP), #0, R1, (R0) | | | | |
| | | | 24 | AB | 9C | AB | D0 | 000C8 | 13\$: | 3743 | |
| | | | 20 | AB | 6E | AB | 000CD | MOVL | -100(CC(B), 36(CC(B) | | |
| | | | 00000000 | EF | 22 | AB | 3C | 000D1 | MOVW | SAVE USZ, 32(CC(B) | |
| | | | 0A | 08 | AB | E8 | 000D9 | MOVZWL | 34(CC(B), RECOUNT | | |
| | | | 7E | 01 | CE | 000DD | BLBS | 8(CC(B), 14\$ | | 3751 | |
| | | | 00000000G | 00 | 01 | FB | 000E0 | MNEGL | #1, -(SP) | | |
| | | | EC | AB | 28 | AB | D0 | 000E7 | 14\$: | 3759 | |
| | | | 5E | 04 | C0 | 000EC | CALLS | #1, BASS\$STOP IO | | | |
| | | | | 3C | BA | 000EF | MOVL | 40(CC(B), -20(CC(B) | | | |
| | | | | 05 | 000F1 | RSB | #4, SP | | | 3761 | |
| | | | | | | POPR | #^M<R2,R3,R4,R5> | | | | |

: Routine Size: 242 bytes, Routine Base: _BASS\$CODE + 0766

: 2490 3762 1

```

2492 3763 1 GLOBAL ROUTINE BASS$REC_GRE ( ! GET (relative) a record
2493 3764 1   FOREIGN_BUFFER, _LOCK_FLAGS) : JSB_DO_READ NOVALUE =
2494 3765 1
2495 3766 1 ++ FUNCTIONAL DESCRIPTION:
2496 3767 1
2497 3768 1   Read one record from a relative file. Modify the RAB if necessary for
2498 3769 1   foreign buffers. Update RECOUNT if successful. Otherwise, signal a fatal error.
2499 3770 1
2500 3771 1
2501 3772 1 FORMAL PARAMETERS:
2502 3773 1
2503 3774 1   FOREIGN_BUFFER.rl.v      The address of the CB of a foreign
2504 3775 1
2505 3776 1   LOCK_FLAGS.rlu.v      buffer or 0
2506 3777 1                           bits to set in the RAB ROP to control
2507 3778 1                           manual record locking (0 if none)
2508 3779 1
2509 3780 1 IMPLICIT INPUTS:
2510 3781 1
2511 3782 1   RAB$W_RSZ           record size for foreign buffer
2512 3783 1   RAB$L_UBF           buffer address for foreign buffer
2513 3784 1
2514 3785 1
2515 3786 1 IMPLICIT OUTPUTS:
2516 3787 1
2517 3788 1   RAB$B_RAC           Record access field
2518 3789 1   RECOUNT            Own storage for RECOUNT function.
2519 3790 1   RAB$L_RBF           Record pointer in RAB.
2520 3791 1   RAB$W_USZ           record size of file buffer
2521 3792 1   RAB$L_UBF           address of buffer for file buffer
2522 3793 1
2523 3794 1
2524 3795 1
2525 3796 1
2526 3797 1
2527 3798 1
2528 3799 1
2529 3800 2
2530 3801 2
2531 3802 2
2532 3803 2
2533 3804 2
2534 3805 2
2535 3806 2
2536 3807 2
2537 3808 2
2538 3809 2
2539 3810 2
2540 3811 2
2541 3812 2
2542 3813 2
2543 3814 2
2544 3815 2
2545 3816 2
2546 3817 2
2547 3818 2
2548 3819 2

1   LOCAL RMS_STATUS,
2   SAVE_USZ:                                ! Save the USZ
3
4   + Save USZ in case it is modified. It is faster to always
5   save and restore it, since that eliminates the test for foreign
6   buffers and single-character mode at the end of this routine.
7
8   - SAVE_USZ = .CCB [RAB$W_USZ];
9
10  !+

```

```
2549      3820 2 | Set the record pointer field in the RAB to the user buffer. This is
2550      3821 2 | done on each element transmission and not just at OPEN because of RMS
2551      3822 2 | Locate mode.
2552      3823 2 | Fill the input buffer with Nulls. Basic semantics require null fill if
2553      3824 2 | the record read is shorter than the buffer.
2554      3825 2 | Set the record access field in the RAB to sequential. Perform the GET.
2555      3826 2 | If RMS returns a failure status, signal the error. If the GET is
2556      3827 2 | successful, then update recount.
2557      3828 2 |
2558      3829 2 |
2559      3830 2 |
2560      3831 2 |
2561      3832 2 |
2562      3833 2 |
2563      3834 3 | There is a foreign buffer. Modify the file buffer to point to the
2564      3835 3 | buffer associated with the foreign buffer's channel.
2565      3836 3 |
2566      3837 3 |
2567      3838 3 | CCB [RAB$L_RBF] = CCB [RAB$L_UBF] = .FOREIGN_BUFFER [RAB$L_UBF];
2568      3839 3 | CCB [RAB$W_RSZ] = CCB [RAB$W_USZ] = .FOREIGN_BUFFER [RAB$W_USZ];
2569      3840 3 | END
2570      3841 3 | ELSE
2571      3842 3 |   CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
2572      3843 3 |   CCB [RAB$B_RAC] = RAB$C_KEY;
2573      3844 3 |
2574      3845 3 |
2575      3846 3 | Set bits in RAB ROP without destroying ULK.
2576      3847 3 |
2577      3848 3 |
2578      3849 3 | CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
2579      3850 3 |
2580      3851 3 | RMS_STATUS = $GET (RAB = .CCB);
2581      3852 3 |
2582      3853 3 | IF .RMS_STATUS EQL RMSS_CONTROLC
2583      3854 3 | THEN
2584      3855 3 |   BAS$SIGNAL_CTRLC ();
2585      3856 3 |
2586      3857 3 |
2587      3858 3 | IF NOT .RMS_STATUS
2588      3859 3 | THEN
2589      3860 3 | BEGIN
2590      3861 3 | We cannot call GET_ERROR because we must restore UBF and USZ.
2591      3862 3 |
2592      3863 3 |
2593      3864 4 | WHILE (.CCB [RAB$L_STS] EQL RMSS_RSA) DO
2594      3865 4 | BEGIN
2595      3866 4 |   SWAIT (RAB = .CCB);
2596      3867 4 |   $GET (RAB = .CCB);
2597      3868 3 | END;
2598      3869 3 |
2599      3870 3 | END;
2600      3871 3 |
2601      3872 2 | Turn off bits in the RAB ROP so that subsequent I/O operations can not
2602      3873 2 | inherit them.
2603      3874 2 |
2604      3875 2 |
2605      3876 2 |
```

```

: 2606      3877 2      CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
: 2607      3878
: 2608      3879
: 2609      3880      !+ If there are no errors, null pad the buffer.
: 2610      3881      !-
: 2611      3882
: 2612      3883      IF (.CCB [RAB$W_USZ] GTR .CCB [RAB$W_RSZ]) AND .CCB [RAB$L_STS]
: 2613      3884      THEN
: 2614      3885          CH$FILL (%X'00',
: 2615      3886          .CCB [RAB$W_USZ] - .CCB [RAB$W_RSZ], .CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
: 2616      3887
: 2617      3888
: 2618      3889      !+ Before checking for errors, restore UBF and USZ, and set RECOUNT.
: 2619      3890      !-
: 2620      3891          CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
: 2621      3892          CCB [RAB$W_USZ] = .SAVE_USZ;
: 2622      3893          RECOUNT = .CCB [RAB$W_RSZ];
: 2623      3894      !+
: 2624      3895          Any error remaining (which will be an error other than Record Stream
: 2625      3896          Active, RSA) is fatal.
: 2626      3897      !-
: 2627      3898
: 2628      3899      IF ( NOT .CCB [RAB$L_STS]) THEN BASS$STOP_IO (BASS$K_IOERR_REC);
: 2629      3900
: 2630      3901      !+
: 2631      3902          Set LUB$A_RBUF_ADR to point to the buffer used by RMS. It may move around
: 2632      3903          due to Locate Mode.
: 2633      3904
: 2634      3905          CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_RBF];
: 2635      3906          RETURN;
: 2636      3907      1      END;                                ! End of BASS$REC_GRE

```

| 3C BB 00000 BASS\$REC GRE:: | | | | | | |
|-----------------------------|----------|----|----|-------------|--------|------------------------|
| | 52 | 20 | 51 | D0 00002 | PUSHR | #^M<R2,R3,R4,R5> |
| | 7E | | AB | 3C 00005 | MOVL | R1, R2 |
| | | | 50 | D5 00009 | MOVZWL | 32(CCB), SAVE_USZ |
| | | | 1A | 13 0000B | TSTL | FOREIGN_BUFFER |
| | 51 | 24 | A0 | D0 0000D | BEQL | 1\$ |
| | 24 | AB | 51 | D0 00011 | MOVL | 36(FOREIGN BUFFER), R1 |
| | 28 | AB | 51 | D0 00015 | MOVL | R1, 36(CCB) |
| | 50 | 20 | A0 | 3C 00019 | MOVZWL | 32(FOREIGN BUFFER), R0 |
| | 20 | AB | 50 | B0 0001D | MOVW | R0, 32(CCB) |
| | 22 | AB | 50 | B0 00021 | MOVW | R0, 34(CCB) |
| | | | 05 | 11 00025 | BRB | 2\$ |
| | 28 | AB | 24 | AB D0 00027 | 1\$: | 36(CCB), 40(CCB) |
| | 1E | AB | 01 | 90 0002C | 2\$: | MOVB #1, 30(CCB) |
| | 04 | AB | 52 | C8 00030 | BISL2 | LOCK_FLAGS, 4(CCB) |
| 00000000G | 00 | | 5B | DD 00034 | PUSHL | CCB |
| 00010651 | 53 | | 01 | FB 00036 | CALLS | #1, SYSSGET |
| | 00010651 | 8F | 50 | D0 0003D | MOVL | R0, RMS_STATUS |
| | | | 53 | D1 00040 | CMPL | RM\$_STATUS, #67153 |
| | | | 07 | 12 00047 | BNEQ | 3\$ |

| | | | | | | | | |
|-----------|-------|----|-------|----------|------|--------|------------------------|------|
| 00000000G | 00 | 00 | FB | 00049 | | CALLS | #0, BASS\$SIGAL_CTRLC | 3855 |
| | 1E | | 53 | E8 00050 | 3\$: | BLBS | RMS STATUS 5\$ | 3857 |
| 000182DA | 8F | 08 | AB | D1 00053 | 4\$: | CMPL | 8(CC(B), #99034 | 3864 |
| | | | 14 | 12 0005B | | BNEQ | 5\$ | |
| | | | 5B | DD 0005D | | PUSHL | CC(B) | 3866 |
| 00000000G | 00 | 01 | FB | 0005F | | CALLS | #1, SYSSWAIT | 3867 |
| | | | 5B | DD 00066 | | PUSHL | CC(B) | |
| 00000000G | 00 | 01 | FB | 00068 | | CALLS | #1, SYSSGET | 3864 |
| | | | E2 | 11 0006F | | BRB | 4\$ | 3877 |
| | 04 AB | 20 | 52 CC | 00071 | 5\$: | XORL2 | LOCK FLAGS, 4(CC(B) | 3883 |
| | 22 AB | | AB | B1 00075 | | CMPW | 32(CC(B), 34(CC(B) | |
| | 19 | 08 | AB | E9 0007C | | BLEQU | 6\$ | |
| | 51 | 20 | AB | 3C 00080 | | BLBC | 8(CC(B), 6\$ | 3886 |
| | 50 | 22 | AB | 3C 00084 | | MOVZWL | 32(CC(B), R1 | |
| | 51 | 50 | C2 | 00088 | | MOVZWL | 34(CC(B), R0 | |
| | 50 | 22 | AB | 3C 0008B | | SUBL2 | R0, R1 | |
| | 50 | 24 | AB | C0 0008F | | MOVZWL | 34(CC(B), R0 | |
| 51 | 00 | 6E | 00 | 2C 00093 | | ADDL2 | 36(CC(B), R0 | |
| | | | 60 | 00098 | | MOVC5 | #0, (SP), #0, R1, (R0) | |
| | 24 AB | 9C | AB | DO 00099 | 6\$: | MOVL | -100(CC(B), 36(CC(B) | 3891 |
| | 20 AB | | 6E | B0 0009E | | MOVW | SAVE USZ, 32(CC(B) | 3892 |
| 00000000' | EF | 22 | AB | 3C 000A2 | | MOVZWL | 34(CC(B), RECOUNT | 3893 |
| | 0A | 08 | AB | E8 000AA | | BLBS | 8(CC(B), 7\$ | 3899 |
| 00000000G | 00 | 7E | 01 | CE 000AE | | MNEGL | #1, -(SP) | |
| | EC AB | 28 | AB | DO 000B1 | | CALLS | #1, BASS\$STOP IO | 3905 |
| | 5E | | 04 | C0 000BD | 7\$: | MOVL | 40(CC(B), -20(CC(B) | 3907 |
| | | | 3C | BA 000C0 | | ADDL2 | #4, SP | |
| | | | 05 | 000C2 | | POPR | #^M<R2,R3,R4,R5> | |
| | | | | | | RSB | | |

: Routine Size: 195 bytes, Routine Base: _BASS\$CODE + 0858

```
2638      3908 1 GLOBAL ROUTINE BASS$REC_GRFA (           ! GET (by RFA) a record
2639          3909 1   FOREIGN_BUFFER, "LOCK_FLAGS) : JSB_DO_READ NOVALUE =
2640          3910 1
2641          3911 1 ++
2642          3912 1   FUNCTIONAL DESCRIPTION:
2643          3913 1
2644          3914 1     Read one record from a file. Modify the RAB if necessary for
2645          3915 1     foreign buffers. Update RECOUNT if successful. Otherwise, signal a fatal error.
2646          3916 1
2647          3917 1   FORMAL PARAMETERS:
2648          3918 1
2649          3919 1     FOREIGN_BUFFER.rl.v           The address of the CB of a foreign
2650          3920 1
2651          3921 1     LOCK_FLAGS.rlu.v           buffer or 0
2652          3922 1
2653          3923 1   IMPLICIT INPUTS:
2654          3924 1
2655          3925 1     RAB$W_RSZ                 record size for foreign buffer
2656          3926 1     RAB$L_UBF                  buffer address for foreign buffer
2657          3927 1
2658          3928 1   IMPLICIT OUTPUTS:
2659          3929 1
2660          3930 1     RAB$B_RAC                 Record access field
2661          3931 1     RECOUNT                   Own storage for RECOUNT function.
2662          3932 1     RAB$L_RBF                 Record pointer in RAB.
2663          3933 1     RAB$W_USZ                 record size of file buffer
2664          3934 1     RAB$L_UBF                  address of buffer for file buffer
2665          3935 1
2666          3936 1   ROUTINE VALUE:
2667          3937 1
2668          3938 1     NONE
2669          3939 1
2670          3940 1   SIDE EFFECTS:
2671          3941 1
2672          3942 1     SIGNALS any RMS errors
2673          3943 1
2674          3944 1
2675          3945 2   BEGIN
2676          3946 2
2677          3947 2   EXTERNAL REGISTER
2678          3948 2     CCB : REF BLOCK [, BYTE];
2679          3949 2
2680          3950 2   MAP
2681          3951 2     FOREIGN_BUFFER : REF BLOCK [, BYTE];
2682          3952 2
2683          3953 2   LOCAL
2684          3954 2     RMS_STATUS,
2685          3955 2     SAVE_USZ;                      ! Save the USZ
2686          3956 2
2687          3957 2
2688          3958 2   +
2689          3959 2     Save USZ in case it is modified. It is faster to always
2690          3960 2     save and restore it, since that eliminates the test for foreign
2691          3961 2     buffers and single-character mode at the end of this routine.
2692          3962 2   -
2693          3963 2     SAVE_USZ = .CCB [RAB$W_USZ];
2694          3964 2   !+
```

```
2695      3965 2 | Set the record pointer field in the RAB to the user buffer. This is
2696      3966 2 | done on each element transmission and not just at OPEN because of RMS
2697      3967 2 | Locate mode.
2698      3968 2 | Fill the input buffer with Nulls. Basic semantics require null fill if
2699      3969 2 | the record read is shorter than the buffer.
2700      3970 2 | Set the record access field in the RAB to sequential. Perform the GET.
2701      3971 2 | If RMS returns a failure status, signal the error. If the GET is
2702      3972 2 | successful, then update recount.
2703      3973 2 |
2704      3974 2 |
2705      3975 2 | IF .FOREIGN_BUFFER NEQ 0
2706      3976 2 | THEN
2707      3977 2 | BEGIN
2708      3978 2 |
2709      3979 2 | There is a foreign buffer. Modify the file buffer to point to the
2710      3980 2 | buffer associated with the foreign buffer's channel.
2711      3981 2 |
2712      3982 2 | CCB [RAB$L_RBF] = CCB [RAB$L_UBF] = .FOREIGN_BUFFER [RAB$L_UBF];
2713      3983 2 | CCB [RAB$W_RSZ] = CCB [RAB$W_USZ] = .FOREIGN_BUFFER [RAB$W_USZ];
2714      3984 2 | END
2715      3985 2 | ELSE
2716      3986 2 | CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
2717      3987 2 |
2718      3988 2 | CCB [RAB$B_RAC] = RAB$C_RFA;
2719      3989 2 |
2720      3990 2 |
2721      3991 2 | Set bits in RAB ROP without destroying ULK.
2722      3992 2 |
2723      3993 2 |
2724      3994 2 | CCB [RAB$L_ROP] = .CCB [RAB$I_ROP] OR .LOCK_FLAGS;
2725      3995 2 |
2726      3996 2 | RMS_STATUS = $GET (RAB = .CCB);
2727      3997 2 |
2728      3998 2 | IF .RMS_STATUS EQL RMSS_CONTROLC
2729      3999 2 | THEN
2730      4000 2 |   BAS$SIGNAL_CTRLC ();
2731      4001 2 |
2732      4002 2 | IF NOT .RMS_STATUS
2733      4003 2 | THEN
2734      4004 2 | BEGIN
2735      4005 2 |
2736      4006 2 | We cannot call GET_ERROR because we must restore UBF and USZ.
2737      4007 2 |
2738      4008 2 |
2739      4009 2 | WHILE (.CCB [RAB$L_STS] EQL RMSS_RSA) DO
2740      4010 2 | BEGIN
2741      4011 2 |   $WAIT (RAB = .CCB);
2742      4012 2 |   $GET (RAB = .CCB);
2743      4013 2 | END;
2744      4014 2 |
2745      4015 2 | END;
2746      4016 2 |
2747      4017 2 |
2748      4018 2 | Turn off bits in the RAB ROP so that subsequent I/O operations can not
2749      4019 2 | inherit them.
2750      4020 2 |
2751      4021 2 |
```

```

2752    4022 2   CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
2753    4023 2
2754    4024 2
2755    4025 2 |+ If there are no errors, null pad the buffer.
2756    4026 2 |-
2757    4027 2
2758    4028 2 IF (.CCB [RAB$W_USZ] GTR .CCB [RAB$W_RSZ]) AND .CCB [RAB$L_STS]
2759    4029 2 THEN
2760    4030 2   CHSFILL (%X'00',
2761    4031 2   .CCB [RAB$W_USZ] - .CCB [RAB$W_RSZ], .CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
2762    4032 2
2763    4033 2 |+
2764    4034 2 | Before checking for errors, restore UBF and USZ, and set RECOUNT.
2765    4035 2 |-
2766    4036 2   CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
2767    4037 2   CCB [RAB$W_USZ] = .SAVE_USZ;
2768    4038 2   RECOUNT = .CCB [RAB$W_RSZ];
2769    4039 2 |+
2770    4040 2 | Any error remaining (which will be an error other than Record Stream
2771    4041 2 | Active, RSA) is fatal.
2772    4042 2 |-
2773    4043 2
2774    4044 2 IF ( NOT .CCB [RAB$L_STS] ) THEN BASS$STOP_IO (BASS$K_IOERR_REC);
2775    4045 2
2776    4046 2 |+
2777    4047 2 | Set LUB$A_RBUF_ADR to point to the buffer used by RMS. It may move around
2778    4048 2 | due to Locate Mode.
2779    4049 2 |-
2780    4050 2   CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_RBF];
2781    4051 2   RETURN;
2782    4052 1   END;

```

! End of BASS\$REC_GRFA

| 3C BB 00000 BASS\$REC GRFA:: | | | | | | |
|------------------------------|----|----|----------------|------------------|------------------------|------|
| | | | PUSHR | #^M<R2,R3,R4,R5> | | 3908 |
| 52 | 7E | 20 | 51 D0 00002 | MOVL | R1, R2 | 3962 |
| | | | 50 AB 3C 00005 | MOVZWL | 32(CC), SAVE_USZ | 3975 |
| | | | 50 D5 00009 | TSTL | FOREIGN_BUFFER | |
| 24 | AB | | 1A 13 0000B | BEQL | 1\$ | |
| 28 | AB | | A0 D0 0000D | MOVL | 36(FOREIGN_BUFFER), R1 | 3982 |
| | | | 51 D0 00011 | MOVL | R1, 36(CC) | |
| | | | 51 D0 00015 | MOVL | R1, 40(CC) | |
| 28 | AB | 24 | A0 3C 00019 | MOVZWL | 32(FOREIGN_BUFFER), R0 | 3983 |
| 20 | AB | | 50 B0 0001D | MOVW | R0, 32(CC) | |
| 22 | AB | | 50 B0 00021 | MOVW | R0, 34(CC) | |
| | | | 05 11 00025 | BRB | 2\$ | 3975 |
| 28 | AB | 24 | AB D0 00027 | MOVL | 36(CC), 40(CC) | 3986 |
| 1E | AB | | 02 90 0002C | MOVB | #2, 30(CC) | 3988 |
| 04 | AB | | 52 C8 00030 | BISL2 | LOCK_FLAGS, 4(CC) | 3994 |
| | | | 5B DD 00034 | PUSHL | CCB | 3996 |
| 00000000G | 00 | | 01 FB 00036 | CALLS | #1, SYSSGET | |
| 00010651 | 53 | | 50 D0 0003D | MOVL | R0, RMS_STATUS | |
| | 8F | | 53 D1 00040 | CMPL | RMS_STATUS, #67153 | 3998 |
| | | | 07 12 00047 | BNEQ | 3\$ | |

| | | | | | | | | | |
|-----------|----|--|-----------|----|----------------|------|--------|------------------------|------|
| 00000000G | 00 | | 00 | FB | 00049 | | CALLS | #0, BASS\$SIGNAL_CTRLC | 4000 |
| 000182DA | 1E | | 08 | 53 | E8 00050 | 3\$: | BLBS | RM\$ STATUS \$S | 4002 |
| | BF | | | AB | D1 00053 | 4\$: | CMPL | 8(CC(B), #99034 | 4009 |
| | | | | 14 | 12 0005B | | BNEQ | 5\$ | |
| | | | | 5B | DD 0005D | | PUSHL | CCB | 4011 |
| 00000000G | 00 | | 01 | FB | 0005F | | CALLS | #1, SYSSWAIT | 4012 |
| 00000000G | 00 | | 5B | DD | 00066 | | PUSHL | CCB | |
| | | | 01 | FB | 00068 | | CALLS | #1, SYSSGET | |
| | | | E2 | 11 | 0006F | | BRB | 4\$ | 4009 |
| 04 | AB | | 52 | CC | 00071 | 5\$: | XORL2 | LOCK FLAGS, 4(CC(B) | 4022 |
| 22 | AB | | 20 | AB | B1 00075 | | CMPW | 32(CC(B), 34(CC(B) | 4028 |
| | | | 1D | 1B | 0007A | | BLEQU | 6\$ | |
| | | | 19 | 08 | AB E9 0007C | | BLBC | 8(CC(B), 6\$ | |
| | | | 51 | 20 | AB 3C 00080 | | MOVZWL | 32(CC(B), R1 | 4031 |
| | | | 50 | 22 | AB 3C 00084 | | MOVZWL | 34(CC(B), R0 | |
| | | | 51 | 50 | C2 00088 | | SUBL2 | R0, R1 | |
| | | | 50 | 22 | AB 3C 0008B | | MOVZWL | 34(CC(B), R0 | |
| | | | 50 | 24 | AB C0 0008F | | ADDL2 | 36(CC(B), R0 | |
| 51 | 00 | | 6E | 00 | 2C 00093 | | MOVC5 | #0, (SP), #0, R1, (R0) | |
| | | | | 60 | 00098 | | | | |
| | | | 24 | AB | 9C AB D0 00099 | 6\$: | MOVL | -100(CC(B), 36(CC(B) | 4036 |
| | | | 20 | AB | 6E B0 0009E | | MOVW | SAVE USZ, 32(CC(B) | 4037 |
| | | | 00000000 | EF | 22 AB 3C 000A2 | | MOVZWL | 34(CC(B), RECOUNT | 4038 |
| | | | 0A | 08 | AB E8 000AA | | BLBS | 8(CC(B), 7\$ | 4044 |
| | | | 00000000G | 7E | 01 CE 000AE | | MNEG | #1, -(SP) | |
| | | | EC | 00 | 01 FB 000B1 | | CALLS | #1, BASS\$\$STOP IO | |
| | | | 5E | AB | 28 AB D0 000B8 | 7\$: | MOVL | 40(CC(B), -20(CC(B) | 4050 |
| | | | | 04 | C0 000BD | | ADDL2 | #4, SP | 4052 |
| | | | | 3C | BA 000C0 | | POPR | #^M<R2,R3,R4,R5> | |
| | | | | 05 | 000C2 | | RSB | | |

; Routine Size: 195 bytes, Routine Base: _BASS\$CODE + 091B

: 2783 4053 1
: 2784 4054 1

```
: 2786      4055 1 GLOBAL ROUTINE BASS$REC_PSE (
: 2787          4056 1     COUNT,
: 2788          4057 1     FOREIGN_BUFFER
: 2789          4058 1 ) : JSB_PUT_NOVALUE =
: 2790          4059 1
: 2791          4060 1 ++
: 2792          4061 1 | FUNCTIONAL DESCRIPTION:
: 2793          4062 1
: 2794          4063 1     Check for "foreign buffers" and point RABSL_RSZ to foreign USZ if there.
: 2795          4064 1     Write one record. If successful then return; otherwise, signal a fatal
: 2796          4065 1     error.
: 2797          4066 1
: 2798          4067 1 | FORMAL PARAMETERS:
: 2799          4068 1
: 2800          4069 1     COUNT.rl.v           No. of bytes to write
: 2801          4070 1     FOREIGN_BUFFER.rl.v   pointer to foreign buffer CB or 0
: 2802          4071 1
: 2803          4072 1 | IMPLICIT INPUTS:
: 2804          4073 1
: 2805          4074 1     RABSW_RSZ            of foreign buffer
: 2806          4075 1     RABSL_RBF             of foreign buffer
: 2807          4076 1     LUB$V_CCO            Cancel control 0
: 2808          4077 1
: 2809          4078 1 | IMPLICIT OUTPUTS:
: 2810          4079 1
: 2811          4080 1     RABSL_RBF            for "file" buffer
: 2812          4081 1     RABSW_RSZ            length of record to write
: 2813          4082 1     LUBSL_LOG_RECNO      logical record number
: 2814          4083 1     RAB$B_RAC             record access field
: 2815          4084 1     RAB$V_CCO            Cancel control 0
: 2816          4085 1
: 2817          4086 1 | ROUTINE VALUE:
: 2818          4087 1
: 2819          4088 1     NONE
: 2820          4089 1
: 2821          4090 1 | SIDE EFFECTS:
: 2822          4091 1
: 2823          4092 1     SIGNALS any RMS errors
: 2824          4093 1 |
: 2825          4094 1
: 2826          4095 2 | BEGIN
: 2827          4096 2
: 2828          4097 2 | EXTERNAL REGISTER
: 2829          4098 2     CCB : REF BLOCK [, BYTE];
: 2830          4099 2
: 2831          4100 2 | LOCAL
: 2832          4101 2     RMS_STATUS;
: 2833          4102 2
: 2834          4103 2 | MAP
: 2835          4104 2     FOREIGN_BUFFER : REF BLOCK [, BYTE];
: 2836          4105 2
: 2837          4106 2
: 2838          4107 2 |+
: 2839          4108 2     Copy the current status of the cancel-control-o bit in the LUB
: 2840          4109 2     (possibly set by RCTRL0) into the RAB, and clear it from the
: 2841          4110 2     LUB. The net effect of this is that if the bit is set in the
: 2842          4111 2     LUB, then the CANCTRL0 modifier will be applied to this write
:                  operation only.
```

```
: 2843      4112 2 !-
: 2844      4113
: 2845      4114 2   CCB [RAB$V_CCO] = CCB [LUB$V_CCO];
: 2846      4115 2   CCB [LUB$V_CCO] = 0;
: 2847      4116
: 2848      4117 2   +
: 2849      4118 2   Set the recordsize field in the RAB based on COUNT.
: 2850      4119 2   Set the record address field in the RAB to the user buffer.
: 2851      4120 2   Perform the PUT.
: 2852      4121 2   If RMS returns a failure status, signal the error.
: 2853      4122 2   -
: 2854      4123
: 2855      4124 2   CCB [RAB$W_RSZ] = .COUNT;
: 2856      4125 2   CCB [RAB$B_RAC] = (IF .CCB [LUB$B_ORGAN] EQL LUB$K_ORG_INDEX THEN RAB$C_KEY ELSE RAB$C_SEQ);
: 2857      4126
: 2858      4127 2   IF .FOREIGN_BUFFER NEQA 0
: 2859      4128 2   THEN
: 2860      4129 2   +
: 2861      4130 2   There is a foreign buffer. Point RAB$L_UBF to it.
: 2862      4131 2   -
: 2863      4132 2   ELSE CCB [RAB$L_RBF] = CCB [RAB$L_UBF] = .FOREIGN_BUFFER [RAB$L_UBF]
: 2864      4133 2   ELSE CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
: 2865      4134
: 2866      4135
: 2867      4136 2   RMS_STATUS = $PUT (RAB = .CCB);
: 2868      4137
: 2869      4138 2   IF .RMS_STATUS EQL RMSS_CONTROLC
: 2870      4139 2   THEN
: 2871      4140 2   BASS$SIGNAL_CTRLC ();
: 2872      4141
: 2873      4142 2   IF NOT .RMS_STATUS
: 2874      4143 2   THEN
: 2875      4144 2   BEGIN
: 2876      4145 2   +
: 2877      4146 2   We cannot call PUT_ERROR because we must restore UBF and USZ.
: 2878      4147 2   -
: 2879      4148 3
: 2880      4149 3   WHILE (.CCB [RAB$L_STS] EQL RMSS_RSA) DO
: 2881      4150 4   BEGIN
: 2882      4151 4   SWAIT (RAB = .CCB);
: 2883      4152 4   SPUT (RAB = .CCB);
: 2884      4153 3   END;
: 2885      4154 3
: 2886      4155 2   END;
: 2887      4156
: 2888      4157 2   +
: 2889      4158 2   Restore RAB$L_UBF in case there was a foreign buffer.
: 2890      4159
: 2891      4160 2   CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
: 2892      4161 2   +
: 2893      4162 2   Point LUB$A_RBUF_PTR off to the buffer used by RMS.
: 2894      4163
: 2895      4164 2   CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_UBF];
: 2896      4165 2   +
: 2897      4166 2   Any error remaining (which will be an error other than Record Stream
: 2898      4167 2   Active, RSA) is fatal.
: 2899      4168 2   -
```

```
; 2900      4169 2
; 2901      4170 2  IF ( NOT .CCB [RAB$L_STS] ) THEN BASS$STOP_IO (BASS$IOERR_REC);
; 2902      4171 2
; 2903      4172 2  RETURN;
; 2904      4173 1  END;
                                         ! End of BASS$REC_PSE
```

| | | | | 5E | 04 C2 00000 BASS\$REC_PSE:: | | | |
|----|----|-----------|-----------|----|-----------------------------|-------|-------------------------|------|
| 07 | 7E | A0 | AB | 01 | 01 02 EF 00003 | SUBL2 | #4, SP | 4055 |
| | | | | | 07 8E F0 00009 | EXTZV | #2, #1, -96(CCB), -(SP) | 4114 |
| | | A0 | AB | | 04 04 8A 0000F | INSV | (SP)+ #7, #1, 7(CCB) | 4115 |
| | | 22 | AB | | 51 B0 00013 | BICB2 | #4, -96(CCB) | 4124 |
| | | | | 03 | AB 91 00017 | MOVW | COUNT, 34(CCB) | 4125 |
| | | | | | 05 12 0001B | CMPB | -60(CCB), #3 | |
| | | | | 51 | 01 D0 0001D | BNEQ | 1\$ | |
| | | | | | 02 11 00020 | MOVL | #1, R1 | |
| | | | | | 51 D4 00022 | BRB | 2\$ | |
| | | 1E | AB | | 51 90 00024 | CLRL | R1 | |
| | | | | | 50 D5 00028 | MOVB | R1, 30(CCB) | 4127 |
| | | | | | 0E 13 0002A | TSTL | FOREIGN_BUFFER | |
| | | | 50 | 24 | A0 D0 0002C | BEQL | 3\$ | |
| | | 24 | AB | | 50 D0 00030 | MOVL | 36(FOREIGN_BUFFER), R0 | 4132 |
| | | 28 | AB | | 50 D0 00034 | MOVL | R0, 36(CCB) | |
| | | | | | 05 11 00038 | MOVL | R0, 40(CCB) | |
| | | 28 | AB | 24 | AB D0 0003A | BRB | 4\$ | |
| | | | | | 5B DD 0003F | MOVL | 36(CCB), 40(CCB) | 4134 |
| | | | 000000006 | 00 | 01 FB 00041 | PUSHL | CCB | 4136 |
| | | | 6E | | 50 D0 00048 | CALLS | #1, SYSSPUT | |
| | | 00010651 | 8F | | 6E D1 0004B | MOVL | R0, RMS_STATUS | 4138 |
| | | | | | 07 12 00052 | CMPL | RMS_STATUS, #67153 | |
| | | 00000000G | 00 | | 00 FB 00054 | BNEQ | 5\$ | |
| | | | 1E | | 6E E8 0005B | CALLS | #0, BASS\$SIGNAL_CTRL_C | 4140 |
| | | 000182DA | 8F | 08 | AB D1 0005E | BLBS | RM\$ STATUS, 7\$ | 4142 |
| | | | | | 14 12 00066 | CMPL | 8(CCB), #99034 | 4149 |
| | | | | | 5B DD 00068 | BNEQ | 7\$ | |
| | | 00000000G | 00 | | 01 FB 0006A | PUSHL | CCB | 4151 |
| | | | | | 5B DD 00071 | CALLS | #1, SYSSWAIT | 4152 |
| | | 00000000G | 00 | | 01 FB 00073 | PUSHL | CCB | |
| | | | | | E2 11 0007A | CALLS | #1, SYSSPUT | |
| | | | 24 | AB | AB D0 0007C | BRB | 6\$ | 4149 |
| | | | EC | AB | 24 AB D0 00081 | MOVL | -100(CCB), 36(CCB) | 4160 |
| | | | 0A | 08 | AB E8 00086 | MOVL | 36(CCB), -20(CCB) | 4164 |
| | | | 7E | | 01 CE 0008A | BLBS | 8(CCB), 8\$ | 4170 |
| | | 00000000G | 00 | | 01 FB 0008D | MNEGL | #1, -(SP) | |
| | | | 5E | | 04 C0 00094 | CALLS | #1, BASS\$STOP_IO | |
| | | | | | 05 00097 | ADDL2 | #4, SP | |
| | | | | | | RSB | | 4173 |

; Routine Size: 152 bytes, Routine Base: _BASS\$CODE + 09DE

; 2905 4174 1

```
; 2907      4175 1 GLOBAL ROUTINE BASS$REC_PRE (
; 2908      4176 1   COUNT,
; 2909      4177 1   FOREIGN_BUFFER
; 2910      4178 1 ) : JSB_PUT-NOVALUE =
; 2911      4179 1
; 2912      4180 1 ++
; 2913      4181 1   FUNCTIONAL DESCRIPTION:
; 2914      4182 1
; 2915      4183 1     Check for a foreign buffer and point to it if necessary.
; 2916      4184 1     Write one record. If successful then return; otherwise, signal a fatal
; 2917      4185 1     error.
; 2918      4186 1
; 2919      4187 1   FORMAL PARAMETERS:
; 2920      4188 1
; 2921      4189 1     COUNT.rl.v          No. of bytes to write
; 2922      4190 1     FOREIGN_BUFFER.rl.v  pointer to foreign CB or 0
; 2923      4191 1
; 2924      4192 1   IMPLICIT INPUTS:
; 2925      4193 1
; 2926      4194 1     RAB$L_UBF          for the foreign buffer (buffer pointer)
; 2927      4195 1     RAB$W_USZ          for the foreign buffer (buffer size)
; 2928      4196 1
; 2929      4197 1   IMPLICIT OUTPUTS:
; 2930      4198 1
; 2931      4199 1     RAB$W_RSZ          length of record to write (file buffer)
; 2932      4200 1     RAB$L_RBF          pointer to file CB
; 2933      4201 1     LUB$L_LOG_RECNO    logical record number
; 2934      4202 1     RAB$B_RAC          record access field
; 2935      4203 1
; 2936      4204 1   ROUTINE VALUE:
; 2937      4205 1
; 2938      4206 1
; 2939      4207 1
; 2940      4208 1
; 2941      4209 1
; 2942      4210 1
; 2943      4211 1
; 2944      4212 1
; 2945      4213 2   BEGIN
; 2946      4214 2
; 2947      4215 2   EXTERNAL REGISTER
; 2948      4216 2     CCB : REF BLOCK [, BYTE];
; 2949      4217 2
; 2950      4218 2
; 2951      4219 2
; 2952      4220 2
; 2953      4221 2
; 2954      4222 2
; 2955      4223 2
; 2956      4224 2
; 2957      4225 2
; 2958      4226 2
; 2959      4227 2
; 2960      4228 2
; 2961      4229 2
; 2962      4230 2
; 2963      4231 2
; 2907      4175 1   LOCAL
; 2908      4176 1     RMS_STATUS;
; 2909      4177 1
; 2910      4178 1   MAP
; 2911      4179 1     FOREIGN_BUFFER : REF BLOCK [, BYTE];
; 2912      4180 1
; 2913      4181 1
; 2914      4182 1
; 2915      4183 1
; 2916      4184 1
; 2917      4185 1
; 2918      4186 1
; 2919      4187 1
; 2920      4188 1
; 2921      4189 1
; 2922      4190 1
; 2923      4191 1
; 2924      4192 1
; 2925      4193 1
; 2926      4194 1
; 2927      4195 1
; 2928      4196 1
; 2929      4197 1
; 2930      4198 1
; 2931      4199 1
; 2932      4200 1
; 2933      4201 1
; 2934      4202 1
; 2935      4203 1
; 2936      4204 1
; 2937      4205 1
; 2938      4206 1
; 2939      4207 1
; 2940      4208 1
; 2941      4209 1
; 2942      4210 1
; 2943      4211 1
; 2944      4212 1
; 2945      4213 2
; 2946      4214 2
; 2947      4215 2
; 2948      4216 2
; 2949      4217 2
; 2950      4218 2
; 2951      4219 2
; 2952      4220 2
; 2953      4221 2
; 2954      4222 2
; 2955      4223 2
; 2956      4224 2
; 2957      4225 2
; 2958      4226 2
; 2959      4227 2
; 2960      4228 2
; 2961      4229 2
; 2962      4230 2
; 2963      4231 2
; 2907      4175 1
; 2908      4176 1
; 2909      4177 1
; 2910      4178 1
; 2911      4179 1
; 2912      4180 1
; 2913      4181 1
; 2914      4182 1
; 2915      4183 1
; 2916      4184 1
; 2917      4185 1
; 2918      4186 1
; 2919      4187 1
; 2920      4188 1
; 2921      4189 1
; 2922      4190 1
; 2923      4191 1
; 2924      4192 1
; 2925      4193 1
; 2926      4194 1
; 2927      4195 1
; 2928      4196 1
; 2929      4197 1
; 2930      4198 1
; 2931      4199 1
; 2932      4200 1
; 2933      4201 1
; 2934      4202 1
; 2935      4203 1
; 2936      4204 1
; 2937      4205 1
; 2938      4206 1
; 2939      4207 1
; 2940      4208 1
; 2941      4209 1
; 2942      4210 1
; 2943      4211 1
; 2944      4212 1
; 2945      4213 2
; 2946      4214 2
; 2947      4215 2
; 2948      4216 2
; 2949      4217 2
; 2950      4218 2
; 2951      4219 2
; 2952      4220 2
; 2953      4221 2
; 2954      4222 2
; 2955      4223 2
; 2956      4224 2
; 2957      4225 2
; 2958      4226 2
; 2959      4227 2
; 2960      4228 2
; 2961      4229 2
; 2962      4230 2
; 2963      4231 2
; 2907      4175 1
; 2908      4176 1
; 2909      4177 1
; 2910      4178 1
; 2911      4179 1
; 2912      4180 1
; 2913      4181 1
; 2914      4182 1
; 2915      4183 1
; 2916      4184 1
; 2917      4185 1
; 2918      4186 1
; 2919      4187 1
; 2920      4188 1
; 2921      4189 1
; 2922      4190 1
; 2923      4191 1
; 2924      4192 1
; 2925      4193 1
; 2926      4194 1
; 2927      4195 1
; 2928      4196 1
; 2929      4197 1
; 2930      4198 1
; 2931      4199 1
; 2932      4200 1
; 2933      4201 1
; 2934      4202 1
; 2935      4203 1
; 2936      4204 1
; 2937      4205 1
; 2938      4206 1
; 2939      4207 1
; 2940      4208 1
; 2941      4209 1
; 2942      4210 1
; 2943      4211 1
; 2944      4212 1
; 2945      4213 2
; 2946      4214 2
; 2947      4215 2
; 2948      4216 2
; 2949      4217 2
; 2950      4218 2
; 2951      4219 2
; 2952      4220 2
; 2953      4221 2
; 2954      4222 2
; 2955      4223 2
; 2956      4224 2
; 2957      4225 2
; 2958      4226 2
; 2959      4227 2
; 2960      4228 2
; 2961      4229 2
; 2962      4230 2
; 2963      4231 2
; 2907      4175 1
; 2908      4176 1
; 2909      4177 1
; 2910      4178 1
; 2911      4179 1
; 2912      4180 1
; 2913      4181 1
; 2914      4182 1
; 2915      4183 1
; 2916      4184 1
; 2917      4185 1
; 2918      4186 1
; 2919      4187 1
; 2920      4188 1
; 2921      4189 1
; 2922      4190 1
; 2923      4191 1
; 2924      4192 1
; 2925      4193 1
; 2926      4194 1
; 2927      4195 1
; 2928      4196 1
; 2929      4197 1
; 2930      4198 1
; 2931      4199 1
; 2932      4200 1
; 2933      4201 1
; 2934      4202 1
; 2935      4203 1
; 2936      4204 1
; 2937      4205 1
; 2938      4206 1
; 2939      4207 1
; 2940      4208 1
; 2941      4209 1
; 2942      4210 1
; 2943      4211 1
; 2944      4212 1
; 2945      4213 2
; 2946      4214 2
; 2947      4215 2
; 2948      4216 2
; 2949      4217 2
; 2950      4218 2
; 2951      4219 2
; 2952      4220 2
; 2953      4221 2
; 2954      4222 2
; 2955      4223 2
; 2956      4224 2
; 2957      4225 2
; 2958      4226 2
; 2959      4227 2
; 2960      4228 2
; 2961      4229 2
; 2962      4230 2
; 2963      4231 2
; 2907      4175 1
; 2908      4176 1
; 2909      4177 1
; 2910      4178 1
; 2911      4179 1
; 2912      4180 1
; 2913      4181 1
; 2914      4182 1
; 2915      4183 1
; 2916      4184 1
; 2917      4185 1
; 2918      4186 1
; 2919      4187 1
; 2920      4188 1
; 2921      4189 1
; 2922      4190 1
; 2923      4191 1
; 2924      4192 1
; 2925      4193 1
; 2926      4194 1
; 2927      4195 1
; 2928      4196 1
; 2929      4197 1
; 2930      4198 1
; 2931      4199 1
; 2932      4200 1
; 2933      4201 1
; 2934      4202 1
; 2935      4203 1
; 2936      4204 1
; 2937      4205 1
; 2938      4206 1
; 2939      4207 1
; 2940      4208 1
; 2941      4209 1
; 2942      4210 1
; 2943      4211 1
; 2944      4212 1
; 2945      4213 2
; 2946      4214 2
; 2947      4215 2
; 2948      4216 2
; 2949      4217 2
; 2950      4218 2
; 2951      4219 2
; 2952      4220 2
; 2953      4221 2
; 2954      4222 2
; 2955      4223 2
; 2956      4224 2
; 2957      4225 2
; 2958      4226 2
; 2959      4227 2
; 2960      4228 2
; 2961      4229 2
; 2962      4230 2
; 2963      4231 2
; 2907      4175 1
; 2908      4176 1
; 2909      4177 1
; 2910      4178 1
; 2911      4179 1
; 2912      4180 1
; 2913      4181 1
; 2914      4182 1
; 2915      4183 1
; 2916      4184 1
; 2917      4185 1
; 2918      4186 1
; 2919      4187 1
; 2920      4188 1
; 2921      4189 1
; 2922      4190 1
; 2923      4191 1
; 2924      4192 1
; 2925      4193 1
; 2926      4194 1
; 2927      4195 1
; 2928      4196 1
; 2929      4197 1
; 2930      4198 1
; 2931      4199 1
; 2932      4200 1
; 2933      4201 1
; 2934      4202 1
; 2935      4203 1
; 2936      4204 1
; 2937      4205 1
; 2938      4206 1
; 2939      4207 1
; 2940      4208 1
; 2941      4209 1
; 2942      4210 1
; 2943      4211 1
; 2944      4212 1
; 2945      4213 2
; 2946      4214 2
; 2947      4215 2
; 2948      4216 2
; 2949      4217 2
; 2950      4218 2
; 2951      4219 2
; 2952      4220 2
; 2953      4221 2
; 2954      4222 2
; 2955      4223 2
; 2956      4224 2
; 2957      4225 2
; 2958      4226 2
; 2959      4227 2
; 2960      4228 2
; 2961      4229 2
; 2962      4230 2
; 2963      4231 2
; 2907      4175 1
; 2908      4176 1
; 2909      4177 1
; 2910      4178 1
; 2911      4179 1
; 2912      4180 1
; 2913      4181 1
; 2914      4182 1
; 2915      4183 1
; 2916      4184 1
; 2917      4185 1
; 2918      4186 1
; 2919      4187 1
; 2920      4188 1
; 2921      4189 1
; 2922      4190 1
; 2923      4191 1
; 2924      4192 1
; 2925      4193 1
; 2926      4194 1
; 2927      4195 1
; 2928      4196 1
; 2929      4197 1
; 2930      4198 1
; 2931      4199 1
; 2932      4200 1
; 2933      4201 1
; 2934      4202 1
; 2935      4203 1
; 2936      4204 1
; 2937      4205 1
; 2938      4206 1
; 2939      4207 1
; 2940      4208 1
; 2941      4209 1
; 2942      4210 1
; 2943      4211 1
; 2944      4212 1
; 2945      4213 2
; 2946      4214 2
; 2947      4215 2
; 2948      4216 2
; 2949      4217 2
; 2950      4218 2
; 2951      4219 2
; 2952      4220 2
; 2953      4221 2
; 2954      4222 2
; 2955      4223 2
; 2956      4224 2
; 2957      4225 2
; 2958      4226 2
; 2959      4227 2
; 2960      4228 2
; 2961      4229 2
; 2962      4230 2
; 2963      4231 2
; 2907      4175 1
; 2908      4176 1
; 2909      4177 1
; 2910      4178 1
; 2911      4179 1
; 2912      4180 1
; 2913      4181 1
; 2914      4182 1
; 2915      4183 1
; 2916      4184 1
; 2917      4185 1
; 2918      4186 1
; 2919      4187 1
; 2920      4188 1
; 2921      4189 1
; 2922      4190 1
; 2923      4191 1
; 2924      4192 1
; 2925      4193 1
; 2926      4194 1
; 2927      4195 1
; 2928      4196 1
; 2929      4197 1
; 2930      4198 1
; 2931      4199 1
; 2932      4200 1
; 2933      4201 1
; 2934      4202 1
; 2935      4203 1
; 2936      4204 1
; 2937      4205 1
; 2938      4206 1
; 2939      4
```

```
2964      4232 2 CCB [RAB$B_RAC] = RABSC_KEY;
2965      4233 2
2966      4234 2 IF .FOREIGN_BUFFER NEQA 0
2967      4235 2 THEN
2968      4236 2+
2969      4237 2 There is a foreign buffer. Point off to the buffer but don't do the
2970      4238 2 size. A PUT with count would not work right, so the size is passed in.
2971      4239 2-
2972      4240 2 CCB [RAB$L_UBF] = CCB [RAB$L_RBF] = .FOREIGN_BUFFER [RAB$L_UBF]
2973      4241 2 ELSE CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
2974      4242 2
2975      4243 2 RMS_STATUS = $PUT (RAB = .CCB);
2976      4244 2
2977      4245 2 IF .RMS_STATUS EQL RMSS_CONTROLC
2978      4246 2 THEN
2979      4247 2     BAS$SIGNAL_CTRLC ();
2980      4248 2
2981      4249 2 IF NOT .RMS_STATUS
2982      4250 2 THEN
2983      4251 2     BEGIN
2984      4252 2+
2985      4253 2 We cannot call GET_ERROR because we must restore UBF and USZ.
2986      4254 2-
2987      4255 2
2988      4256 2
2989      4257 2 WHILE (.CCB [RAB$L_STS] EQL RMSS_RSA) DO
2990      4258 2     BEGIN
2991      4259 2     SWAIT (RAB = .CCB);
2992      4260 2     $PUT (RAB = .CCB);
2993      4261 2     END;
2994      4262 2
2995      4263 2     END;
2996      4264 2
2997      4265 2+
2998      4266 2 Restore RAB$L_UBF in case there was a foreign buffer.
2999      4267 2-
3000      4268 2     CCB [RAB$L_UBF] = .CCB [LUBSA_UBF];
3001      4269 2+
3002      4270 2 Point LUBSA_BUF_PTR off to the buffer used by RMS.
3003      4271 2-
3004      4272 2     CCB [LUBSA_RBUF_ADR] = .CCB [RAB$L_UBF];
3005      4273 2+
3006      4274 2 Any error remaining (which will be an error other than Record Stream
3007      4275 2 Active, RSA) is fatal.
3008      4276 2-
3009      4277 2
3010      4278 2 IF ( NOT .CCB [RAB$L_STS]) THEN BAS$STOP_IO (BASSK_IOERR_REC);
3011      4279 2
3012      4280 2 RETURN;
3013      4281 1 END;
```

! End of BASS\$REC_PRE

| | | | | | | | |
|-----------|----|----|----|----------|------------|------------------------|--------|
| 22 | AB | | 51 | B0 00003 | MOVW | COUNT 34(CC8) | : 4231 |
| 1E | AB | | 01 | 90 00007 | MOV8 | #1, 30(CC8) | : 4232 |
| | | | 50 | D5 0000B | TSTL | FOREIGN_BUFFER | : 4234 |
| | | | 0E | 13 0000D | BEQL | 1S | |
| 28 | 50 | 24 | A0 | D0 0000F | MOVL | 36(FOREIGN_BUFFER), R0 | : 4240 |
| 24 | AB | | 50 | D0 00013 | MOVL | R0, 40(CC8) | |
| | | | 50 | D0 00017 | MOVL | R0, 36(CC8) | |
| 28 | AB | 24 | 05 | 11 0001B | BRB | 2S | |
| | | | AB | D0 0001D | 1\$: MOVL | 36(CC8), 40(CC8) | : 4242 |
| 00000000G | 00 | | 5B | DD 00022 | 2\$: PUSHL | CC8 | : 4244 |
| | | | 01 | FB 00024 | CALLS | #1, SY\$PUT | |
| 00010651 | 6E | | 50 | D0 0002B | MOVL | R0, RMS STATUS | |
| | 8F | | 6E | D1 0002E | CMPL | RMS_STATUS, #67153 | : 4246 |
| 00000000G | 00 | | 07 | 12 00035 | BNEQ | 3S | |
| | | | 00 | FB 00037 | CALLS | #0, BASS\$SIGNAL_CTRLC | : 4248 |
| 000182DA | 1E | 08 | 6E | E8 0003E | 3\$: BLBS | RMS_STATUS, 5S | : 4250 |
| | 8F | | AB | D1 00041 | 4\$: CMPL | 8(CC8), #99034 | : 4257 |
| | | | 14 | 12 00049 | BNEQ | 5S | |
| 00000000G | 00 | | 5B | DD 0004B | PUSHL | CC8 | : 4259 |
| | | | 01 | FB 0004D | CALLS | #1, SY\$WAIT | |
| 00000000G | 00 | | 5B | DD 00054 | PUSHL | CC8 | : 4260 |
| | | | 01 | FB 00056 | CALLS | #1, SY\$PUT | |
| | | | E2 | 11 0005D | BRB | 4S | : 4257 |
| 24 | AB | 9C | AB | D0 0005F | 5\$: MOVL | -100(CC8), 36(CC8) | : 4268 |
| EC | AB | 24 | AB | D0 00064 | MOVL | 36(CC8), -20(CC8) | : 4272 |
| 0A | 08 | AB | E8 | 00069 | BLBS | 8(CC8), 6S | : 4278 |
| 00000000G | 7E | | 01 | CE 0006D | MNEG | #1, -(SP) | |
| | 00 | | 01 | FB 00070 | CALLS | #1, BASS\$STOP_IO | |
| | 5E | | 04 | C0 00077 | 6\$: ADDL2 | #4, SP | |
| | | | 05 | 0007A | RSB | | : 4281 |

; Routine Size: 123 bytes, Routine Base: _BASS\$CODE + 0A76

; 3014 4282 1

```
3016      4283 1 GLOBAL ROUTINE BASS$REC_FSE (           ! FIND (sequential) a record
3017          4284 1   LOCK_FLAGS
3018          4285 1   ) : JSB_REC2 NOVALUE =
3019
3020          4287 1 ++
3021          4288 1 | FUNCTIONAL DESCRIPTION:
3022
3023          4290 1 | Find next record. If successful then return; otherwise, signal a fatal
3024          4291 1 | error.
3025
3026          4292 1 | FORMAL PARAMETERS:
3027          4293 1 |
3028          4295 1 |   LOCK_FLAGS.rlu.v     bits to set in the RAB ROP to control manual
3029          4296 1 |   record locking (0 if none)
3030
3031          4297 1 | IMPLICIT INPUTS:
3032          4299 1 |   NONE
3033          4300 1 |
3034          4301 1 | IMPLICIT OUTPUTS:
3035          4302 1 |   RAB$B_RAC       record access field
3036          4303 1 |
3037          4304 1 | ROUTINE VALUE:
3038          4305 1 |   NONE
3039          4306 1 |
3040          4307 1 | SIDE EFFECTS:
3041          4308 1 |
3042          4309 1 |
3043          4310 1 |   Finds next record in file on this logical unit.
3044          4311 1 |
3045          4312 1 |   SIGNALS any RMS errors
3046          4313 1 |
3047          4314 1 | --
3048          4315 1 | BEGIN
3049          4316 2 |
3050          4317 2 | EXTERNAL REGISTER
3051          4318 2 |   CCB : REF BLOCK [, BYTE];
3052          4319 2 |
3053          4320 2 |
3054          4321 2 | LOCAL
3055          4322 2 |   RMS_STATUS;
3056          4323 2 |
3057          4324 2 | +
3058          4325 2 |   Set the record access field in the RAB to sequential. Perform the FIND.
3059          4326 2 |   If RMS returns a failure status, signal the error.
3060          4327 2 |
3061          4328 2 |
3062          4329 2 |   CCB [RAB$B_RAC] = RAB$C_SEQ;
3063          4330 2 |
3064          4331 2 | +
3065          4332 2 |   Set bits in RAB ROP without clearing ULK.
3066          4333 2 |
3067          4334 2 |
3068          4335 2 |   CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
3069          4336 2 |
3070          4337 2 | +
3071          4338 2 |   perform the FIND.
3072          4339 2 | -
```

```

3073    4340  2
3074    4341  2 RMS_STATUS = $FIND (RAB = .CCB);
3075    4342  2
3076    4343  2
3077    4344  2
3078    4345  2
3079    4346  2
3080    4347  2 CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
3081    4348  2
3082    4349  2
3083    4350  2
3084    4351  2
3085    4352  2 IF NOT .RMS_STATUS
3086    4353  2 THEN
3087    4354  2     BASS$STOP_IO (BASS$IOERR_REC);
3088    4355  2
3089    4356  2 RETURN;
3090    4357  1 END;           ! End of BASS$REC_FSE

```

.EXTRN SY\$FIND

| | | 52 DD 00000 BASS\$REC FSE:: | | | |
|--|--|-----------------------------|-------|--------------------|------|
| | | | PUSHL | R2 | 4283 |
| | | | MOVL | R0, R2 | 4329 |
| | | | CLRB | 30(CC8) | 4335 |
| | | | BISL2 | LOCK_FLAGS, 4(CC8) | 4341 |
| | | | PUSHL | CC8 | 4347 |
| | | | CALLS | #1, SY\$FIND | 4352 |
| | | | XORL2 | LOCK_FLAGS, 4(CC8) | 4354 |
| | | | BLBS | RMS_STATUS, 1\$ | 4357 |
| | | | MNEGL | #1, -(SP) | |
| | | | CALLS | #1, BASS\$STOP_IO | |
| | | | POPR | #^M<R2> | |
| | | | RSB | | |

; Routine Size: 41 bytes, Routine Base: _BASS\$CODE + 0AF1

```
4358 1 GLOBAL ROUTINE BASS$REC_FRFA (           ! FIND (by RFA) a record
4359 1   LOCK_FLAGS
4360 1   ) : JSB_REC2 NOVALUE =
4361 1
4362 1 ++
4363 1 FUNCTIONAL DESCRIPTION:
4364 1
4365 1   Find record by RFA stored in the RAB. If successful then return; otherwise, signal a fatal
4366 1   error.
4367 1
4368 1 FORMAL PARAMETERS:
4369 1
4370 1   LOCK_FLAGS.rlu.v      bits to set in the RAB ROP to control manual
4371 1   record locking (0 if none)
4372 1
4373 1 IMPLICIT INPUTS:
4374 1   NONE
4375 1
4376 1 IMPLICIT OUTPUTS:
4377 1
4378 1   RAB$B_RAC          record access field
4379 1
4380 1 ROUTINE VALUE:
4381 1   NONE
4382 1
4383 1 SIDE EFFECTS:
4384 1
4385 1   Finds next record in file on this logical unit.
4386 1
4387 1   SIGNALS any RMS errors
4388 1
4389 1 --
4390 1
4391 2 BEGIN
4392 2
4393 2 EXTERNAL REGISTER
4394 2   CCB : REF BLOCK [, BYTE];
4395 2
4396 2 LOCAL
4397 2   RMS_STATUS;
4398 2
4399 2
4400 2   Set the record access field in the RAB to sequential. Perform the FIND.
4401 2   If RMS returns a failure status, signal the error.
4402 2
4403 2
4404 2   CCB [RAB$B_RAC] = RAB$C_RFA;
4405 2
4406 2
4407 2   Set bits in RAB ROP without clearing ULK.
4408 2
4409 2
4410 2   CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
4411 2
4412 2
4413 2   perform the FIND.
4414 2
```

```

: 3149      4415 2
: 3150      4416 2 RMS_STATUS = $FIND (RAB = .CCB);
: 3151      4417 2
: 3152      4418 2 |+
: 3153      4419 2 | Turn off bits so that subsequent operations will not inherit them.
: 3154      4420 2 |-
: 3155      4421 2
: 3156      4422 2 CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
: 3157      4423 2
: 3158      4424 2 |+
: 3159      4425 2 | Signal if the FIND failed.
: 3160      4426 2 |-
: 3161      4427 2 IF NOT .RMS_STATUS
: 3162      4428 2 THEN
: 3163      4429 2   BASS$STOP_IO (BASSK_IOERR_REC);
: 3164      4430 2
: 3165      4431 2 RETURN;
: 3166      4432 1 END;                                ! End of BASS$REC_FRFA

```

| | | 52 DD 00000 BASS\$REC_FRFA:: | | |
|--|--------------|------------------------------|--------------------------|------|
| | | PUSHL R2 | | 4358 |
| | 1E AB | 50 D0 00002 | MOVL R0, R2 | 4404 |
| | 04 AB | 02 90 00005 | MOVBL #2, 30(CCB) | 4410 |
| | 00000000G 00 | 52 C8 00009 | BISL2 LOCK_FLAGS, 4(CCB) | 4416 |
| | 04 AB | 5B DD 0000D | PUSHL CCB | 4422 |
| | 0A | 01 FB 0000F | CALLS #1, SY\$FIND | 4427 |
| | 00000000G 00 | 52 CC 00016 | XORL2 LOCK_FLAGS, 4(CCB) | 4429 |
| | 7E | 50 E8 0001A | BLBS RMS_STATUS, 1\$ | |
| | 01 | 01 CE 0001D | MNEGL #1, -(SP) | |
| | 04 BA 00020 | 01 FB 00020 | CALLS #1, BASS\$STOP_IO | |
| | 05 00029 | 04 BA 00027 1\$: | POPR #^M<R2> | |
| | | 05 00029 | RSB | 4432 |

: Routine Size: 42 bytes, Routine Base: _BASS\$CODE + 0B1A

: 3167 4433 1
: 3168 4434 1

```
3170      4435 1 GLOBAL ROUTINE BASS$REC_FRE (           ! FIND (relative) a record
3171          4436 1   LOCK_FLAGS
3172          4437 1   ) : JSB_REC0 NOVALUE =
3173          4438 1
3174          4439 1 ++
3175          4440 1   FUNCTIONAL DESCRIPTION:
3176          4441 1
3177          4442 1   Find next record. If successful then return; otherwise, signal a fatal
3178          4443 1   error.
3179          4444 1
3180          4445 1   FORMAL PARAMETERS:
3181          4446 1
3182          4447 1   LOCK_FLAGS.rlu.v     bits to set in the RAB ROP to control manual
3183          4448 1   record locking (0 if none)
3184          4449 1
3185          4450 1   IMPLICIT INPUTS:
3186          4451 1   NONE
3187          4452 1
3188          4453 1   IMPLICIT OUTPUTS:
3189          4454 1   RAB$B_RAC       record access field
3190          4455 1
3191          4456 1   ROUTINE VALUE:
3192          4457 1   NONE
3193          4458 1
3194          4459 1   SIDE EFFECTS:
3195          4460 1
3196          4461 1
3197          4462 1
3198          4463 1
3199          4464 1   SIGNALS any RMS errors
3200          4465 1
3201          4466 1
3202          4467 2   BEGIN
3203          4468 2
3204          4469 2   EXTERNAL REGISTER
3205          4470 2   [CB : REF BLOCK [, BYTE];
3206          4471 2
3207          4472 2
3208          4473 2   LOCAL
3209          4474 2   RMS_STATUS;
3210          4475 2
3211          4476 2
3212          4477 2
3213          4478 2
3214          4479 2
3215          4480 2
3216          4481 2
3217          4482 2
3218          4483 2
3219          4484 2
3220          4485 2
3221          4486 2
3222          4487 2
3223          4488 2
3224          4489 2
3225          4490 2
3226          4491 2
+
|+ Set the record access field in the RAB to sequential. Perform the FIND.
|- If RMS returns a failure status, signal the error.
+
|+ CCB [RAB$B_RAC] = RAB$C_KEY;
|- Set bits in the RAB ROP without clearing ULK.
+
|+ CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
|- perform the FIND.
```

```

: 3227    4492 2 RMS_STATUS = $FIND (RAB = .CCB);
: 3228    4493 2
: 3229    4494 2
: 3230    4495 2 !+
: 3231    4496 2 Turn off bits in the RAB ROP so that subsequent operations do not
: 3232    4497 2 inherit them.
: 3233    4498 2 !-
: 3234    4499 2 CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
: 3235    4500 2
: 3236    4501 2 !+
: 3237    4502 2 Signal if the FIND failed.
: 3238    4503 2 !-
: 3239    4504 2 IF NOT .RMS_STATUS
: 3240    4505 2 THEN
: 3241    4506 2     BASS$STOP_IO (BASS$K_IOERR_REC);
: 3242    4507 2
: 3243    4508 2 RETURN;
: 3244    4509 1 END;

```

! End of BAS\$\$REC_FRE

| | | | | |
|-----------|----|----|---------------------------|--------------------------|
| 1E | AB | 01 | 90 00000 BAS\$\$REC_FRE:: | |
| 04 | AB | 04 | AE C8 00004 | MOV B #1, 30(CC8) |
| 00000000G | 00 | | 5B DD 00009 | BISL2 LOCK_FLAGS, 4(CC8) |
| 04 | AB | 04 | 01 FB 0000B | PUSHL CCB |
| | 0A | | AE CC 00012 | CALLS #1, SYS\$FIND |
| | 7E | | 50 E8 00017 | XORL2 LOCK_FLAGS, 4(CC8) |
| 00000000G | 00 | | 01 CE 0001A | BLBS RMS_STATUS, 1\$ |
| | | | 01 FB 0001D | MNEGL #1, -(SP) |
| | | | 05 00024 1\$: | CALLS #1, BASS\$STOP_IO |
| | | | | RSB |

; Routine Size: 37 bytes, Routine Base: _BASS\$CODE + 0B44

; 3245 4510 1

: 4480
: 4486
: 4492
: 4499
: 4504
: 4506
: 4509

```

3247 4511 1 GLOBAL ROUTINE BASS$REC_FIN (
3248 4512 1 KEY_NO, REL_OP, KEY, LOCK_FLAGS) : JSB_REC_IND1 NOVALUE =
3249 4513 1 ! FIND (indexed) a record
3250 4514 1 ++
3251 4515 1 FUNCTIONAL DESCRIPTION:
3252 4516 1
3253 4517 1 Find indicated record. If successful then return; otherwise, signal a fatal
3254 4518 1 error.
3255 4519 1
3256 4520 1 FORMAL PARAMETERS:
3257 4521 1
3258 4522 1 KEY_NO.rl.v key of reference
3259 4523 1 REL_OP.rl.v relational operator for key
3260 4524 1 KEY.rt.dx key to search for
3261 4525 1 LOCK_FLAGS.rlu.v bits to set in the RAB ROP to control
3262 4526 1 manual record locking (0 if none)
3263 4527 1
3264 4528 1 IMPLICIT INPUTS:
3265 4529 1
3266 4530 1 NONE
3267 4531 1
3268 4532 1 IMPLICIT OUTPUTS:
3269 4533 1
3270 4534 1 RAB$L_KBF pointer to the desired key value
3271 4535 1 RAB$B_KSZ size of desired key value
3272 4536 1 RAB$M_KGE relational in RAB$L_ROP indicating greater than
3273 4537 1 or equal
3274 4538 1 RAB$M_KGT relational in RAB$L_ROP indicating greater than
3275 4539 1 RAB$B_KRF indicates key of reference
3276 4540 1 RAB$B_RAC record access field
3277 4541 1
3278 4542 1 ROUTINE VALUE:
3279 4543 1
3280 4544 1 NONE
3281 4545 1
3282 4546 1 SIDE EFFECTS:
3283 4547 1
3284 4548 1 See RMS Reference manual for discussion on whether match will be exact,
3285 4549 1 generic, approximate, or generic-approximate.
3286 4550 1 SIGNALS any RMS errors
3287 4551 1 --
3288 4552 1
3289 4553 2 BEGIN
3290 4554 2
3291 4555 2 EXTERNAL REGISTER
3292 4556 2 CCB : REF_BLOCK [, BYTE];
3293 4557 2
3294 4558 2 MAP
3295 4559 2 KEY : REF_BLOCK [8, BYTE];
3296 4560 2
3297 4561 2 LITERAL
3298 4562 2 K_EQUAL = 0, ! search for key equal
3299 4563 2 K_GREATER_EQUAL = 1, ! search for key GEQ
3300 4564 2 K_GREATER_THAN = 2; ! search for key GTR
3301 4565 2
3302 4566 2 LOCAL
3303 4567 2 RMS_STATUS;

```

```
: 3304      4568 2
: 3305      4569 2
: 3306      4570 2
: 3307      4571 2
: 3308      4572 2
: 3309      4573 2
: 3310      4574 2
: 3311      4575 2
: 3312      4576 2
: 3313      4577 2
: 3314      4578 2
: 3315      4579 2
: 3316      4580 2
: 3317      4581 2
: 3318      4582 2
: 3319      4583 2
: 3320      4584 2
: 3321      4585 2
: 3322      4586 2
: 3323      4587 2
: 3324      4588 2
: 3325      4589 2
: 3326      4590 2
: 3327      4591 2
: 3328      4592 3
: 3329      4593 3
: 3330      4594 3
: 3331      4595 2
: 3332      4596 2
: 3333      4597 2
: 3334      4598 3
: 3335      4599 3
: 3336      4600 3
: 3337      4601 2
: 3338      4602 2
: 3339      4603 2
: 3340      4604 2
: 3341      4605 2
: 3342      4606 2
: 3343      4607 2
: 3344      4608 2
: 3345      4609 2
: 3346      4610 2
: 3347      4611 2
: 3348      4612 2
: 3349      4613 2
: 3350      4614 2
: 3351      4615 2
: 3352      4616 2
: 3353      4617 2
: 3354      4618 2
: 3355      4619 2
: 3356      4620 2
: 3357      4621 2
: 3358      4622 2
: 3359      4623 2
: 3360      4624 2

+ Set the key buffer field, the key size field, the key of reference,
and the relational bits in the ROP.
Set the record access field in the RAB to key. Perform the FIND.
If RMS returns a failure status, signal the error.
-
CCB [RAB$B_RAC] = RAB$C_KEY;
CCB [RAB$L_KBF] = .KEY [DSC$A_POINTER];
CCB [RAB$B_KRF] = .KEY_NO;
CCB [RAB$B_KSZ] = (IF .KEY [DSC$B_DTYPE] NEQ DSC$K_DTYPE_P
THEN
    .KEY [DSC$W_LENGTH]
ELSE
    (.KEY [DSC$W_LENGTH]/2 + 1));
CASE .REL_OP FROM K_EQUAL TO K_GREATER_THAN OF
SET
[K_EQUAL] :
    CCB [RAB$V_KGE] = CCB [RAB$V_KGT] = 0;
[K_GREATER_EQUAL] :
    BEGIN
        CCB [RAB$V_KGE] = 1;
        CCB [RAB$V_KGT] = 0;
    END;
[K_GREATER_THAN] :
    BEGIN
        CCB [RAB$V_KGT] = 1;
        CCB [RAB$V_KGE] = 0;
    END;
TES;

+ Set bits in the RAB ROP without clearing ULK.
-
CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
+
| perform the FIND.
-
RMS_STATUS = $FIND (RAB = .CCB);
+
| Turn off bits in the RAB ROP so that subsequent operations do not
inherit them.
-
CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
+
| Signal if the FIND failed.
```

```
: 3361    4625  2      :-  

.: 3362    4626  2      IF NOT .RMS_STATUS  

.: 3363    4627  2      THEN  

.: 3364    4628  2      BASS$STOP_IO (BASSK_IOERR_REC);  

.: 3365    4629  2      RETURN;  

.: 3366    4630  2      END;
```

! End of BASS\$REC_FIN

| 52 DD 00000 BASS\$REC FIN:: | | | | | | | | | | |
|-----------------------------|----|-------|----|-----------|-------------|--------|-------------------|------------|--|------|
| 1E | AB | | 04 | 01 | 90 00002 | PUSHL | R2 | | | 4511 |
| 30 | AB | | | A2 | D0 00006 | MOVB | #1, 30(CCB) | | | 4576 |
| 35 | AB | | 02 | 50 | 90 0000B | MOVL | 4(KEY), 48(CCB) | | | 4577 |
| | 15 | | | A2 | 91 0000F | MOVB | KEY NO, 53(CCB) | | | 4578 |
| | | | | 05 | 13 00013 | CMPB | 2(KEY), #21 | | | 4579 |
| | | | | 62 | 3C 00015 | BEQL | 1\$ | | | 4581 |
| | | | | 08 | 11 00018 | MOVZWL | (KEY), R2 | | | 4583 |
| | | | | 62 | 3C 0001A | BRB | 2\$ | | | |
| | | | | 02 | C6 0001D | MOVZWL | (KEY), R2 | | | |
| | | | | 52 | D6 00020 | DIVL2 | #2, R2 | | | |
| | | | | 52 | 90 00022 | INCL | R2 | | | |
| | | | 34 | AB | AB 00026 | MOVB | R2, 52(CCB) | | | 4579 |
| | | | 02 | 00 | 51 CF 0002A | MOVAB | 4(CCB), R2 | | | 4589 |
| 0018 | | 0000D | | 04 | 0006 | CASEL | REL_OP, #0, #2 | | | 4585 |
| | | | | | 0002E | .WORD | 4\$-3\$,- | | | |
| | | | | | | | 5\$-3\$,- | | | |
| | | | | 02 | A2 40 | 6S- | 6\$-3\$ | | | |
| | | | | | 8F 10 | BICB2 | #64, 2(R2) | | | 4589 |
| | | | | 02 | A2 20 | 7S- | BRB | | | 4593 |
| | | | | 02 | A2 40 | 5\$: | BISB2 | #32, 2(R2) | | 4594 |
| | | | | 02 | A2 09 | BRB | #64, 2(R2) | | | 4585 |
| | | | | 02 | A2 8F | 8S- | BISB2 | #64, 2(R2) | | 4599 |
| | | | | 02 | A2 20 | 6\$: | BICB2 | #32, 2(R2) | | 4600 |
| | | | | 62 | 53 C8 0004F | BISL2 | LOCK_FLAGS, (R2) | | | 4608 |
| | | | | | 5B DD 00052 | PUSHL | CCB | | | 4614 |
| | | | | 00000000G | 00 01 | CALLS | #1, SYSS\$FIND | | | |
| | | | | 62 | FB 00054 | XORL2 | LOCK_FLAGS, (R2) | | | 4621 |
| | | | | 0A | 53 CC 0005B | BLBS | RMS_STATUS, 9\$ | | | 4626 |
| | | | | 7E | 50 E8 0005E | MNEGL | #1, -(SP) | | | 4628 |
| | | | | 00000000G | 00 01 | CALLS | #1, BASS\$STOP_IO | | | |
| | | | | | FB 00064 | POPR | #^M<R2> | | | |
| | | | | | BA 0006B | RSB | | | | 4631 |

: Routine Size: 110 bytes, Routine Base: _BASS\$CODE + 0B69

: 3368 4632 1

```

: 3370      4633 1 GLOBAL ROUTINE BASS$REC_DSE          ! DELETE (sequential) a record
: 3371      4634 1 : JSB_REC0 NOVALUE =
: 3372      4635 1 ++
: 3373      4636 1 FUNCTIONAL DESCRIPTION:
: 3374      4637 1 Delete current record. If successful then return; otherwise, signal a fatal
: 3375      4638 1 error.
: 3376      4639 1 FORMAL PARAMETERS:
: 3377      4640 1
: 3378      4641 1 NONE
: 3379      4642 1 IMPLICIT INPUTS:
: 3380      4643 1
: 3381      4644 1 NONE
: 3382      4645 1 IMPLICIT OUTPUTS:
: 3383      4646 1
: 3384      4647 1 RAB$B_RAC           record access field
: 3385      4648 1 ROUTINE VALUE:
: 3386      4649 1
: 3387      4650 1 NONE
: 3388      4651 1 SIDE EFFECTS:
: 3389      4652 1
: 3390      4653 1 -- SIGNALS any RMS errors
: 3391      4654 1
: 3392      4655 1 BEGIN
: 3393      4656 1
: 3394      4657 1 EXTERNAL REGISTER
: 3395      4658 1   CCB : REF BLOCK [, BYTE];
: 3396      4659 1
: 3397      4660 1 ++
: 3398      4661 1 Set the record access field in the RAB to sequential. Perform the DELETE.
: 3399      4662 1
: 3400      4663 2
: 3401      4664 2
: 3402      4665 2
: 3403      4666 2
: 3404      4667 2
: 3405      4668 2
: 3406      4669 2
: 3407      4670 2
: 3408      4671 2
: 3409      4672 2
: 3410      4673 2
: 3411      4674 2
: 3412      4675 2
: 3413      4676 2
: 3414      4677 2
: 3415      4678 1
: 3416      4679 1
: 3417      4680 1
: 3418      4681 1
: 3419      4682 1
: 3420      4683 1
: 3421      4684 1
: 3422      4685 1
: 3423      4686 1
: 3424      4687 1
: 3425      4688 1
: 3426      4689 1
: 3427      4690 1
: 3428      4691 1
: 3429      4692 1
: 3430      4693 1
: 3431      4694 1
: 3432      4695 1
: 3433      4696 1
: 3434      4697 1
: 3435      4698 1
: 3436      4699 1
: 3437      4700 1
: 3438      4701 1
: 3439      4702 1
: 3440      4703 1
: 3441      4704 1
: 3442      4705 1
: 3443      4706 1
: 3444      4707 1
: 3445      4708 1
: 3446      4709 1
: 3447      4710 1
: 3448      4711 1
: 3449      4712 1
: 3450      4713 1
: 3451      4714 1
: 3452      4715 1
: 3453      4716 1
: 3454      4717 1
: 3455      4718 1
: 3456      4719 1
: 3457      4720 1
: 3458      4721 1
: 3459      4722 1
: 3460      4723 1
: 3461      4724 1
: 3462      4725 1
: 3463      4726 1
: 3464      4727 1
: 3465      4728 1
: 3466      4729 1
: 3467      4730 1
: 3468      4731 1
: 3469      4732 1
: 3470      4733 1
: 3471      4734 1
: 3472      4735 1
: 3473      4736 1
: 3474      4737 1
: 3475      4738 1
: 3476      4739 1
: 3477      4740 1
: 3478      4741 1
: 3479      4742 1
: 3480      4743 1
: 3481      4744 1
: 3482      4745 1
: 3483      4746 1
: 3484      4747 1
: 3485      4748 1
: 3486      4749 1
: 3487      4750 1
: 3488      4751 1
: 3489      4752 1
: 3490      4753 1
: 3491      4754 1
: 3492      4755 1
: 3493      4756 1
: 3494      4757 1
: 3495      4758 1
: 3496      4759 1
: 3497      4760 1
: 3498      4761 1
: 3499      4762 1
: 3500      4763 1
: 3501      4764 1
: 3502      4765 1
: 3503      4766 1
: 3504      4767 1
: 3505      4768 1
: 3506      4769 1
: 3507      4770 1
: 3508      4771 1
: 3509      4772 1
: 3510      4773 1
: 3511      4774 1
: 3512      4775 1
: 3513      4776 1
: 3514      4777 1
: 3515      4778 1
: 3516      4779 1
: 3517      4780 1
: 3518      4781 1
: 3519      4782 1
: 3520      4783 1
: 3521      4784 1
: 3522      4785 1
: 3523      4786 1
: 3524      4787 1
: 3525      4788 1
: 3526      4789 1
: 3527      4790 1
: 3528      4791 1
: 3529      4792 1
: 3530      4793 1
: 3531      4794 1
: 3532      4795 1
: 3533      4796 1
: 3534      4797 1
: 3535      4798 1
: 3536      4799 1
: 3537      4800 1
: 3538      4801 1
: 3539      4802 1
: 3540      4803 1
: 3541      4804 1
: 3542      4805 1
: 3543      4806 1
: 3544      4807 1
: 3545      4808 1
: 3546      4809 1
: 3547      4810 1
: 3548      4811 1
: 3549      4812 1
: 3550      4813 1
: 3551      4814 1
: 3552      4815 1
: 3553      4816 1
: 3554      4817 1
: 3555      4818 1
: 3556      4819 1
: 3557      4820 1
: 3558      4821 1
: 3559      4822 1
: 3560      4823 1
: 3561      4824 1
: 3562      4825 1
: 3563      4826 1
: 3564      4827 1
: 3565      4828 1
: 3566      4829 1
: 3567      4830 1
: 3568      4831 1
: 3569      4832 1
: 3570      4833 1
: 3571      4834 1
: 3572      4835 1
: 3573      4836 1
: 3574      4837 1
: 3575      4838 1
: 3576      4839 1
: 3577      4840 1
: 3578      4841 1
: 3579      4842 1
: 3580      4843 1
: 3581      4844 1
: 3582      4845 1
: 3583      4846 1
: 3584      4847 1
: 3585      4848 1
: 3586      4849 1
: 3587      4850 1
: 3588      4851 1
: 3589      4852 1
: 3590      4853 1
: 3591      4854 1
: 3592      4855 1
: 3593      4856 1
: 3594      4857 1
: 3595      4858 1
: 3596      4859 1
: 3597      4860 1
: 3598      4861 1
: 3599      4862 1
: 3600      4863 1
: 3601      4864 1
: 3602      4865 1
: 3603      4866 1
: 3604      4867 1
: 3605      4868 1
: 3606      4869 1
: 3607      4870 1
: 3608      4871 1
: 3609      4872 1
: 3610      4873 1
: 3611      4874 1
: 3612      4875 1
: 3613      4876 1
: 3614      4877 1
: 3615      4878 1
: 3616      4879 1
: 3617      4880 1
: 3618      4881 1
: 3619      4882 1
: 3620      4883 1
: 3621      4884 1
: 3622      4885 1
: 3623      4886 1
: 3624      4887 1
: 3625      4888 1
: 3626      4889 1
: 3627      4890 1
: 3628      4891 1
: 3629      4892 1
: 3630      4893 1
: 3631      4894 1
: 3632      4895 1
: 3633      4896 1
: 3634      4897 1
: 3635      4898 1
: 3636      4899 1
: 3637      4900 1
: 3638      4901 1
: 3639      4902 1
: 3640      4903 1
: 3641      4904 1
: 3642      4905 1
: 3643      4906 1
: 3644      4907 1
: 3645      4908 1
: 3646      4909 1
: 3647      4910 1
: 3648      4911 1
: 3649      4912 1
: 3650      4913 1
: 3651      4914 1
: 3652      4915 1
: 3653      4916 1
: 3654      4917 1
: 3655      4918 1
: 3656      4919 1
: 3657      4920 1
: 3658      4921 1
: 3659      4922 1
: 3660      4923 1
: 3661      4924 1
: 3662      4925 1
: 3663      4926 1
: 3664      4927 1
: 3665      4928 1
: 3666      4929 1
: 3667      4930 1
: 3668      4931 1
: 3669      4932 1
: 3670      4933 1
: 3671      4934 1
: 3672      4935 1
: 3673      4936 1
: 3674      4937 1
: 3675      4938 1
: 3676      4939 1
: 3677      4940 1
: 3678      4941 1
: 3679      4942 1
: 3680      4943 1
: 3681      4944 1
: 3682      4945 1
: 3683      4946 1
: 3684      4947 1
: 3685      4948 1
: 3686      4949 1
: 3687      4950 1
: 3688      4951 1
: 3689      4952 1
: 3690      4953 1
: 3691      4954 1
: 3692      4955 1
: 3693      4956 1
: 3694      4957 1
: 3695      4958 1
: 3696      4959 1
: 3697      4960 1
: 3698      4961 1
: 3699      4962 1
: 3700      4963 1
: 3701      4964 1
: 3702      4965 1
: 3703      4966 1
: 3704      4967 1
: 3705      4968 1
: 3706      4969 1
: 3707      4970 1
: 3708      4971 1
: 3709      4972 1
: 3710      4973 1
: 3711      4974 1
: 3712      4975 1
: 3713      4976 1
: 3714      4977 1
: 3715      4978 1
: 3716      4979 1
: 3717      4980 1
: 3718      4981 1
: 3719      4982 1
: 3720      4983 1
: 3721      4984 1
: 3722      4985 1
: 3723      4986 1
: 3724      4987 1
: 3725      4988 1
: 3726      4989 1
: 3727      4990 1
: 3728      4991 1
: 3729      4992 1
: 3730      4993 1
: 3731      4994 1
: 3732      4995 1
: 3733      4996 1
: 3734      4997 1
: 3735      4998 1
: 3736      4999 1
: 3737      5000 1
: 3738      5001 1
: 3739      5002 1
: 3740      5003 1
: 3741      5004 1
: 3742      5005 1
: 3743      5006 1
: 3744      5007 1
: 3745      5008 1
: 3746      5009 1
: 3747      5010 1
: 3748      5011 1
: 3749      5012 1
: 3750      5013 1
: 3751      5014 1
: 3752      5015 1
: 3753      5016 1
: 3754      5017 1
: 3755      5018 1
: 3756      5019 1
: 3757      5020 1
: 3758      5021 1
: 3759      5022 1
: 3760      5023 1
: 3761      5024 1
: 3762      5025 1
: 3763      5026 1
: 3764      5027 1
: 3765      5028 1
: 3766      5029 1
: 3767      5030 1
: 3768      5031 1
: 3769      5032 1
: 3770      5033 1
: 3771      5034 1
: 3772      5035 1
: 3773      5036 1
: 3774      5037 1
: 3775      5038 1
: 3776      5039 1
: 3777      5040 1
: 3778      5041 1
: 3779      5042 1
: 3780      5043 1
: 3781      5044 1
: 3782      5045 1
: 3783      5046 1
: 3784      5047 1
: 3785      5048 1
: 3786      5049 1
: 3787      5050 1
: 3788      5051 1
: 3789      5052 1
: 3790      5053 1
: 3791      5054 1
: 3792      5055 1
: 3793      5056 1
: 3794      5057 1
: 3795      5058 1
: 3796      5059 1
: 3797      5060 1
: 3798      5061 1
: 3799      5062 1
: 3800      5063 1
: 3801      5064 1
: 3802      5065 1
: 3803      5066 1
: 3804      5067 1
: 3805      5068 1
: 3806      5069 1
: 3807      5070 1
: 3808      5071 1
: 3809      5072 1
: 3810      5073 1
: 3811      5074 1
: 3812      5075 1
: 3813      5076 1
: 3814      5077 1
: 3815      5078 1
: 3816      5079 1
: 3817      5080 1
: 3818      5081 1
: 3819      5082 1
: 3820      5083 1
: 3821      5084 1
: 3822      5085 1
: 3823      5086 1
: 3824      5087 1
: 3825      5088 1
: 3826      5089 1
: 3827      5090 1
: 3828      5091 1
: 3829      5092 1
: 3830      5093 1
: 3831      5094 1
: 3832      5095 1
: 3833      5096 1
: 3834      5097 1
: 3835      5098 1
: 3836      5099 1
: 3837      5100 1
: 3838      5101 1
: 3839      5102 1
: 3840      5103 1
: 3841      5104 1
: 3842      5105 1
: 3843      5106 1
: 3844      5107 1
: 3845      5108 1
: 3846      5109 1
: 3847      5110 1
: 3848      5111 1
: 3849      5112 1
: 3850      5113 1
: 3851      5114 1
: 3852      5115 1
: 3853      5116 1
: 3854      5117 1
: 3855      5118 1
: 3856      5119 1
: 3857      5120 1
: 3858      5121 1
: 3859      5122 1
: 3860      5123 1
: 3861      5124 1
: 3862      5125 1
: 3863      5126 1
: 3864      5127 1
: 3
```

BASS\$REC_PROC
1-095

G 11
16-Sep-1984 01:01:12 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:56:35 [BASRTL.SRC]BASREC[PRO.B32;1]

Page 92
(34)

00000000G 00 01 FB 00012 CALLS #1, BASS\$STOP_IO
05 00019 1\$: RSB

; 4678

; Routine Size: 26 bytes, Routine Base: _BASS\$CODE + 0BD7

; 3416 4679 1

```
3418      4680 1 GLOBAL ROUTINE BASS$REC_UNL          ! UNLOCK a record
3419          4681 1 : JSB_REC0 NOVALUE =
3420          4682 1
3421          4683 1 ++
3422          4684 1 FUNCTIONAL DESCRIPTION:
3423          4685 1
3424          4686 1     Unlock the current record. If successful or no records locked,
3425          4687 1     then return; otherwise, signal a fatal error.
3426          4688 1
3427          4689 1 FORMAL PARAMETERS:
3428          4690 1
3429          4691 1     NONE
3430          4692 1
3431          4693 1 IMPLICIT INPUTS:
3432          4694 1
3433          4695 1     NONE
3434          4696 1
3435          4697 1 IMPLICIT OUTPUTS:
3436          4698 1
3437          4699 1     RAB$B_RAC           record access field
3438          4700 1
3439          4701 1 ROUTINE VALUE:
3440          4702 1
3441          4703 1     NONE
3442          4704 1
3443          4705 1 SIDE EFFECTS:
3444          4706 1
3445          4707 1     SIGNALs any RMS errors
3446          4708 1
3447          4709 1
3448          4710 2 BEGIN
3449          4711 2
3450          4712 2 EXTERNAL REGISTER
3451          4713 2     CCB : REF BLOCK [, BYTE];
3452          4714 2
3453          4715 2
3454          4716 2     +
3455          4717 2     | Set the record access field in the RAB to sequential. Perform the UNLOCK.
3456          4718 2     | If RMS returns a failure status, signal the error.
3457          4719 2     |
3458          4720 2     CCB [RAB$B_RAC] = RAB$C_SEQ;
3459          4721 2
3460          4722 2     IF NOT $RELEASE (RAB = .(CCB)
3461          4723 2     THEN
3462          4724 2
3463          4725 2     IF .CCB [RAB$L_STS] NEQ RMSS_RNL
3464          4726 2     THEN
3465          4727 2
3466          4728 2     +
3467          4729 2     | An error was returned, check for 'record not locked'.
3468          4730 2
3469          4731 2
3470          4732 2     RETURN:
3471          4733 1     END;                      ! End of BASS$REC_UNL
```

.EXTRN SYSSRELEASE

| | | | | | |
|-----------|----|----|------------------------------------|--|--------|
| | 1E | AB | 94 00000 BASS\$REC UNL:: | | |
| | | 5B | DD 00003 CLRB 30(CCB) | | : 4720 |
| 00000000G | 00 | 01 | FB 00005 PUSHL CCB | | : 4722 |
| | 14 | 50 | E8 0000C CALLS #1, SYSSRELEASE | | |
| 000181A0 | 8F | 08 | AB D1 0000F BLBS R0, 1\$ | | : 4725 |
| | | 0A | 13 00017 CMPL 8(CCB), #98720 | | |
| 00000000G | 00 | 01 | CE 00019 BEQL 1\$ | | : 4730 |
| | 7E | 01 | FB 0001C MNEGL #1, -(SP) | | |
| | | 05 | 00023 1\$: CALLS #1, BASS\$STOP_IO | | |
| | | | RSB | | : 4733 |

; Routine Size: 36 bytes. Routine Base: _BAS\$CODE + 0BF1

: 3472 4734 1

```
3474      4735 1 GLOBAL ROUTINE BASS$REC_FEE           ! FREE all locked records
3475      4736 1 : JSB_REC0 NOVALUE =
3476      4737 1 ++
3477      4738 1 FUNCTIONAL DESCRIPTION:
3478      4739 1
3479      4740 1     Free all locked records. If successful or no records locked,
3480      4741 1     then return; otherwise, signal a fatal error.
3481      4742 1
3482      4743 1 FORMAL PARAMETERS:
3483      4744 1
3484      4745 1     NONE
3485      4746 1
3486      4747 1 IMPLICIT INPUTS:
3487      4748 1
3488      4749 1     NONE
3489      4750 1
3490      4751 1 IMPLICIT OUTPUTS:
3491      4752 1
3492      4753 1     RAB$B_RAC          record access field
3493      4754 1
3494      4755 1 ROUTINE VALUE:
3495      4756 1
3496      4757 1     NONE
3497      4758 1
3498      4759 1 SIDE EFFECTS:
3499      4760 1
3500      4761 1     SIGNALS any RMS errors
3501      4762 1
3502      4763 1
3503      4764 1
3504      4765 2 BEGIN
3505      4766 2
3506      4767 2 EXTERNAL REGISTER
3507      4768 2     CCB : REF BLOCK [, BYTE];
3508      4769 2
3509      4770 2
3510      4771 2     +
3511      4772 2     Set the record access field in the RAB to sequential. Perform the FREE.
3512      4773 2     If RMS returns a failure status, signal the error.
3513      4774 2
3514      4775 2
3515      4776 2
3516      4777 3 IF NOT $FREE (RAB = .CCB)
3517      4778 2 THEN
3518      4779 2
3519      4780 2     IF .CCB [RAB$L_STS] NEQ RMSS_RNL
3520      4781 2     THEN
3521      4782 2
3522      4783 2     +
3523      4784 2     An error was returned, check for "record not locked".
3524      4785 2     -
3525      4786 2     BASS$STOP_IO (BASS$K_IOERR_REC);
3526      4787 2 RETURN;
3527      4788 1 END;                                ! End of BASS$REC_FEE
```

.EXTRN SYSS\$FREE

| | | | | | |
|-----------|----|----|--------------------------|---------------------------|--------|
| | 1E | AB | 94 00000 BASS\$REC_FEE:: | | |
| | | | CLRB 30(CCB) | | : 4775 |
| 00000000G | 00 | 5B | DD 00003 | PUSHL CCB | : 4777 |
| | 14 | 01 | FB 00005 | CALLS #1, SYSS\$FREE | |
| 000181A0 | 8F | 50 | E8 0000C | BLBS R0, 1\$ | |
| | | 08 | AB D1 0000F | CMPL 8(CCB), #98720 | : 4780 |
| | | | 0A 13 00017 | BEQL 1\$ | |
| 00000000G | 7E | 01 | CE 00019 | MNEG L #1, -(SP) | : 4785 |
| | 00 | 01 | FB 0001C | CALLS #1, BASS\$\$STOP_IO | |
| | | 05 | 00023 1\$: RSB | | : 4788 |

; Routine Size: 36 bytes, Routine Base: _BASS\$CODE + 0C15

; 3528 4789 1

```
: 3530      4790 1 GLOBAL ROUTINE BASS$REC_UPD (
: 3531          4791 1     COUNT
: 3532          4792 1     ) : JSB_DO_WRITE NOVALUE =
: 3533          4793 1
: 3534          4794 1     ++
: 3535          4795 1     FUNCTIONAL DESCRIPTION:
: 3536          4796 1
: 3537          4797 1     Update current record. If successful then return; otherwise, signal a fatal
: 3538          4798 1     error.
: 3539          4799 1
: 3540          4800 1     FORMAL PARAMETERS:
: 3541          4801 1
: 3542          4802 1     COUNT.rl.v           No. of bytes in record to update
: 3543          4803 1
: 3544          4804 1     IMPLICIT INPUTS:
: 3545          4805 1
: 3546          4806 1     NONE
: 3547          4807 1
: 3548          4808 1     IMPLICIT OUTPUTS:
: 3549          4809 1
: 3550          4810 1     RAB$B_RAC           record access field
: 3551          4811 1     RAB$W_RSZ            record size
: 3552          4812 1
: 3553          4813 1     ROUTINE VALUE:
: 3554          4814 1
: 3555          4815 1     NONE
: 3556          4816 1
: 3557          4817 1     SIDE EFFECTS:
: 3558          4818 1
: 3559          4819 1     Update current record in file on this logical unit.
: 3560          4820 1     SIGNALs any RMS errors
: 3561          4821 1     --
: 3562          4822 1     BEGIN
: 3563          4823 2
: 3564          4824 2
: 3565          4825 2     EXTERNAL REGISTER
: 3566          4826 2     CCB : REF BLOCK [. , BYTE];
: 3567          4827 2
: 3568          4828 2     +
: 3569          4829 2     Point RBF to the user buffer.
: 3570          4830 2     Set the record access field in the RAB to sequential. Perform the UPDATE.
: 3571          4831 2     If RMS returns a failure status, signal the error.
: 3572          4832 2     -
: 3573          4833 2     CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
: 3574          4834 2     CCB [RAB$W_RSZ] = .COUNT;
: 3575          4835 2     CCB [RAB$B_RAC] = RAB$C_SEQ;
: 3576          4836 2
: 3577          4837 2     IF NOT $UPDATE (RAB = .CCB) THEN BASS$STOP_IO (BASSK_ICERR_REC);
: 3578          4838 2
: 3579          4839 2     +
: 3580          4840 2     Point LUBSA_RBUF_ADR to the buffer used by RMS for MOVE.
: 3581          4841 2     -
: 3582          4842 2     CCB [LUBSA_RBUF_ADR] = .CCB [RAB$L_UBF];
: 3583          4843 2     RETURN;
: 3584          4844 1     END;                                ! End of BASS$REC_UPD
```

.EXTRN SYSSUPDATE

| | | |
|--------------|--------------------------------|--------------------------|
| 28 AB | 24 AB DO 00000 BASS\$REC_UPD:: | MOVL 36(CC(B) 40(CC(B) |
| 22 AB | 1E AB 50 B0 00005 | MOVW COUNT 34(CC(B) |
| | 5B DD 0000C | CLRB 30(CC(B) |
| 00000000G 00 | 01 FB 0000E | PUSHL CCB |
| 0A | 50 E8 00015 | CALLS #1, SYSSUPDATE |
| 7E | 01 CE 00018 | BLBS R0, 1\$ |
| 00000000G 00 | 01 FB 0001B | MNEGL #1, -(SP) |
| EC AB | 24 AB DO 00022 1\$: 05 00027 | CALLS #1, BASS\$STOP IO |
| | | MOVL 36(CC(B), -20(CC(B) |
| | | RSB |

: 4833
: 4834
: 4835
: 4837
: 4842
: 4844

; Routine Size: 40 bytes. Routine Base: _BASS\$CODE + 0C39

; 3585 4845 1

```

: 3587   4846 1 GLOBAL ROUTINE BASS$REC_RSE           ! RESTORE (sequential) to beginning of file
: 3588   4847 1 : JSB_REC0 NOVALUE ≡
: 3589   4848 1 ++
: 3590   4849 1 FUNCTIONAL DESCRIPTION:
: 3591   4850 1     Rewind the file. If successful then return; otherwise, signal a fatal
: 3592   4851 1     error.
: 3593   4852 1 FORMAL PARAMETERS:
: 3594   4853 1     NONE
: 3595   4854 1 IMPLICIT INPUTS:
: 3596   4855 1     NONE
: 3597   4856 1 IMPLICIT OUTPUTS:
: 3598   4857 1     RAB$B_RAC          record access field
: 3599   4858 1 ROUTINE VALUE:
: 3600   4859 1     NONE
: 3601   4860 1 SIDE EFFECTS:
: 3602   4861 1     SIGNALS any RMS errors
: 3603   4862 1 --
: 3604   4863 1     BEGIN
: 3605   4864 1     EXTERNAL REGISTER
: 3606   4865 1     CCB : REF BLOCK [, BYTE];
: 3607   4866 1
: 3608   4867 1     +
: 3609   4868 1     Set the record access field in the RAB to sequential. Perform the REWIND.
: 3610   4869 1     If RMS returns a failure status, signal the error.
: 3611   4870 1     -
: 3612   4871 1     CCB [RAB$B_RAC] = RAB$C_SEQ;
: 3613   4872 1     IF NOT $REWIND (RAB = .CCB) THEN BASS$STOP_IO (BASS$K_IOERR_REC);
: 3614   4873 1     RETURN;
: 3615   4874 1     END;                                ! End of BASS$REC_RSE
: 3616   4875 1
: 3617   4876 2
: 3618   4877 2
: 3619   4878 2
: 3620   4879 2
: 3621   4880 2
: 3622   4881 2
: 3623   4882 2
: 3624   4883 2
: 3625   4884 2
: 3626   4885 2
: 3627   4886 2
: 3628   4887 2
: 3629   4888 2
: 3630   4889 2
: 3631   4890 2
: 3632   4891 1

```

.EXTRN SYSSREWIND

| | | | |
|-------------|--------------------------------|-------|----------------|
| | 1E AB 94 00000 BASS\$REC_RSE:: | | |
| | 5B DD 00003 | CLRB | 30(CCB) |
| 0000000G 00 | 01 FB 00005 | PUSHL | CCB |
| 0A | 50 E8 0000C | CALLS | #1. SYSSREWIND |
| 7E | 01 CE 0000F | BLBS | R0: 1\$ |
| | | MNEG | #1, -(SP) |

4886
4888

BASS\$REC_PROC
1-095

B 12
16-Sep-1984 01:01:12 VAX-11 Bliss-32 v4.0-742
14-Sep-1984 11:56:35 [BASRTL.SRC]BASRECPRO.B32;1

Page 100
(38)

00000000G 00 01 FB 00012 CALLS #1, BASS\$STOP_10
 05 00019 1\$: RSB

; 4891

; Routine Size: 26 bytes, Routine Base: _BASS\$CODE + 0C61

; 3633 4892 1

```

: 3635      4893 1 GLOBAL ROUTINE BASS$REC_RIN (
: 3636      4894 1           KEY_NO) : JSB_REC_IND NOVALUE =
: 3637      4895 1
: 3638      4896 1
: 3639      4897 1     ++ FUNCTIONAL DESCRIPTION:
: 3640      4898 1
: 3641      4899 1           Rewind the file. If successful then return; otherwise, signal a fatal
: 3642      4900 1           error.
: 3643      4901 1
: 3644      4902 1     FORMAL PARAMETERS:
: 3645      4903 1
: 3646      4904 1           KEY_NO.rl.v          key of reference
: 3647      4905 1
: 3648      4906 1     IMPLICIT INPUTS:
: 3649      4907 1           NONE
: 3650      4908 1
: 3651      4909 1
: 3652      4910 1
: 3653      4911 1
: 3654      4912 1
: 3655      4913 1           RAB$B_KRF          key of reference
: 3656      4914 1           RAB$B_RAC         record access field
: 3657      4915 1
: 3658      4916 1     ROUTINE VALUE:
: 3659      4917 1           NONE
: 3660      4918 1
: 3661      4919 1     SIDE EFFECTS:
: 3662      4920 1
: 3663      4921 1           SIGNALs any RMS errors
: 3664      4922 1
: 3665      4923 1
: 3666      4924 2     BEGIN
: 3667      4925 2
: 3668      4926 2
: 3669      4927 2     EXTERNAL REGISTER
: 3670      4928 2           CCB : REF BLOCK [, BYTE];
: 3671      4929 2
: 3672      4930 2
: 3673      4931 2     + Set the key of reference
: 3674      4932 2           Set the record access field in the RAB to key. Perform the REWIND.
: 3675      4933 2           If RMS returns a failure status, signal the error.
: 3676      4934 2
: 3677      4935 2           CCB [RAB$B_KRF] = .KEY_NO;
: 3678      4936 2           CCB [RAB$B_RAC] = RAB$C_KEY;
: 3679      4937 2
: 3680      4938 2     IF NOT $REWIND (RAB = .CCB) THEN BASS$STOP_IO (BASS$K_IOERR_REC);
: 3681      4939 2
: 3682      4940 2     RETURN;
: 3683      4941 1           END;           ! End of BASS$REC_RIN

```

| | | |
|-------|-----------------------------|-----------------------|
| 35 AB | 50 90 00000 BASS\$REC_RIN:: | |
| 1E AB | 01 90 00004 | MOV B KEY_NO, 53(CCB) |
| | | MOV B #1,-30(CCB) |

: 4935
: 4936

| | | |
|--------------|---------------|-----------------------|
| 00000000G 00 | 5B DD 00008 | PUSHL CCB |
| 0A | 01 FB 0000A | CALLS #1, SYSSREWIND |
| 7E | 50 E8 00011 | BLBS R0, 1\$ |
| 00000000G 00 | 01 CE 00014 | MNEGL #1, -(SP) |
| | 01 FB 00017 | CALLS #1, BASSSTOP_10 |
| | 05 0001E 1\$: | RSB |

: 4938

: 4941

: Routine Size: 31 bytes, Routine Base: _BASSCODE + 0C7B

: 3684 4942 1

```

3686      4943 1 GLOBAL ROUTINE BASS$REC_SSE           ! SCRATCH (sequential) a record
3687      4944 1 : JSB_REC0 NOVALUE =
3688      4945 1
3689      4946 1 ++
3690      4947 1 FUNCTIONAL DESCRIPTION:
3691      4948 1
3692      4949 1     Truncate this file. If successful then return; otherwise, signal a fatal
3693      4950 1     error.
3694      4951 1
3695      4952 1 FORMAL PARAMETERS:
3696      4953 1
3697      4954 1     NONE
3698      4955 1
3699      4956 1 IMPLICIT INPUTS:
3700      4957 1
3701      4958 1     NONE
3702      4959 1
3703      4960 1 IMPLICIT OUTPUTS:
3704      4961 1
3705      4962 1     RAB$B_RAC          record access field
3706      4963 1
3707      4964 1 ROUTINE VALUE:
3708      4965 1
3709      4966 1     NONE
3710      4967 1
3711      4968 1 SIDE EFFECTS:
3712      4969 1
3713      4970 1     SIGNALs any RMS errors
3714      4971 1
3715      4972 1
3716      4973 2 BEGIN
3717      4974 2
3718      4975 2 EXTERNAL REGISTER
3719      4976 2     CCB : REF BLOCK [, BYTE];
3720      4977 2
3721      4978 2
3722      4979 2     !+ Set the record access field in the RAB to sequential. Perform the TRUNCATE.
3723      4980 2     !+ If RMS returns a failure status, signal the error.
3724      4981 2     !-
3725      4982 2
3726      4983 2     CCB [RAB$B_RAC] = RAB$C_SEQ;
3727      4984 2
3728      4985 2     IF NOT $TRUNCATE (RAB = .(CCB) THEN BASS$STOP_IO (BASS$IOERR_REC);
3729      4986 2
3730      4987 2     RETURN;
3731      4988 1     END:           ! End of BASS$REC_SSE

```

.EXTRN SYS\$TRUNCATE

| | | | | |
|-----------|----|--------------------------|-------|-------------------|
| 1E | AB | 94 00000 BASS\$REC_SSE:: | | |
| 00000000G | 00 | | CLRB | 30(CCB) |
| 0A | | | PUSHL | CCB |
| 7E | | | CALLS | #1, SYS\$TRUNCATE |
| | | | BLBS | R0, 1\$ |
| | | | MNEGL | #1, -(SP) |

: 4983
: 4985

BASS\$REC_PROC
1-095

F 12
16-Sep-1984 01:01:12 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:56:35 [BASRTL.SRC]BASREC[PRO.B32;1]

Page 104
(40)

00000000G 00 01 FB 00012 CALLS #1, BASS\$STOP_IO
 05 00019 1\$: RSB

; 4988

; Routine Size: 26 bytes, Routine Base: _BASS\$CODE + 0C9A

; 3732 4989 1

```
3734      4990 1 ROUTINE PUT_ERROR (
3735          4991 1     SIGNAL_OR_STOP
3736          4992 1     ) : CALL_CCB NOVALUE =
3737          4993 1
3738          4994 1 ++ FUNCTIONAL DESCRIPTION:
3739          4995 1
3740          4996 1
3741          4997 1     Here on $PUT errors, check for Record stream active error (RMSS_RSA)
3742          4998 1     If this error, WAIT until not active and try $PUT again.
3743          4999 1     This recovers from AST I/O which can occur out of the middle
3744          5000 1     of synchronous I/O at non-AST level.
3745          5001 1
3746          5002 1 CALLING SEQUENCE:
3747          5003 1
3748          5004 1     PUT_ERROR (signal_or_stop)
3749          5005 1
3750          5006 1 FORMAL PARAMETERS:
3751          5007 1
3752          5008 1     SIGNAL_OR_STOP.rl.v           whether to signal or stop
3753          5009 1
3754          5010 1 IMPLICIT INPUTS:
3755          5011 1
3756          5012 1     CCB                      Addr. of current LUB/ISB/RAB
3757          5013 1
3758          5014 1 IMPLICIT OUTPUTS:
3759          5015 1
3760          5016 1     LUB$V_OUTBUF_DR           Cleared to indicate clean buffer
3761          5017 1
3762          5018 1 ROUTINE VALUE:
3763          5019 1
3764          5020 1     NONE
3765          5021 1
3766          5022 1 SIDE EFFECTS:
3767          5023 1
3768          5024 1     SWAITs and then tries $PUT again, until success or any error
3769          5025 1     except record stream active.
3770          5026 1 --
3771          5027 1
3772          5028 2 BEGIN
3773          5029 2
3774          5030 2 EXTERNAL REGISTER
3775          5031 2     CCB : REF BLOCK [, BYTE];
3776          5032 2
3777          5033 2 WHILE .CCB [RAB$L_STS] EQL RMSS_RSA DO
3778          5034 3     BEGIN
3779          5035 3     SWAIT (RAB = .CCB);
3780          5036 4     $PUT (RAB = .CCB)
3781          5037 2     END;
3782          5038 2
3783          5039 2 IF NOT .CCB [RAB$L_STS]
3784          5040 2 THEN
3785          5041 3     BEGIN
3786          5042 3
3787          5043 3     Clear the buffer dirty bit so if there is anything there BAS$CLOSE won't
3788          5044 3     get confused, and try to do another PUT.
3789          5045 3
3790          5046 3     CCB [LUB$V_OUTBUF_DR] = 0;
```

```

3791    5047 3
3792    5048 3
3793    5049 3
3794    5050 3
3795    5051 3
3796    5052 3
3797    5053 3
3798    5054 2
3799    5055 2
3800    5056 1

      IF .SIGNAL_OR_STOP EQL K_SIGNAL
      THEN   BAS$$SIGNAL_IO (BASSK_IOERR_REC)
      ELSE   BAS$$STOP_IO (BASSK_IOERR_REC);
      END;
      RETURN;
      END;

```

: End of PUT_ERROR

0000 00000 PUT_ERROR:

| | | | | | | WORD | Save nothing | |
|-----------|----|----|-------------|------|-------|----------------------|--------------|------|
| 000182DA | BF | 08 | AB D1 00002 | 1\$: | Cmpl | 8(CC(B), #99034 | | 4990 |
| | | | 14 12 0000A | | BNEQ | 2\$ | | 5033 |
| 00000000G | 00 | | 5B DD 0000C | | PUSHL | CC(B | | 5035 |
| 00000000G | 00 | | 01 FB 0000E | | CALLS | #1, SYSSWAIT | | 5036 |
| | | | 5B DD 00015 | | PUSHL | CC(B | | 5039 |
| | | | 01 FB 00017 | | CALLS | #1, SYSSPUT | | 5046 |
| | | | E2 11 0001E | | BRB | 1\$ | | 5048 |
| | FE | 08 | AB E8 00020 | 2\$: | BLBS | 8(CC(B), 4\$ | | 5050 |
| | AB | | 08 8A 00024 | | BICB2 | #8, -2(CC(B) | | 5052 |
| | 01 | 04 | AC D1 00028 | | Cmpl | SIGNAL_OR_STOP, #1 | | 5056 |
| | | | 0B 12 0002C | | BNEQ | 3\$ | | |
| 00000000G | 00 | | 01 CE 0002E | | MNEGL | #1, -(SP) | | |
| | | | 01 FB 00031 | | CALLS | #1, BAS\$\$SIGNAL_IO | | |
| | | | 04 00038 | | RET | | | |
| 00000000G | 00 | | 01 CE 00039 | 3\$: | MNEGL | #1, -(SP) | | |
| | | | 01 FB 0003C | | CALLS | #1, BAS\$\$STOP_IO | | |
| | | | 04 00043 | 4\$: | RET | | | |

; Routine Size: 68 bytes, Routine Base: _BASS\$CODE + 0CB4

```
3802      5057 1 ROUTINE GET_ERROR (
3803      5058 1   SIGNAL_OR_STOP
3804      5059 1   ) : CALL_CCB NOVALUE =
3805      5060 1
3806      5061 1 ++
3807      5062 1   FUNCTIONAL DESCRIPTION:
3808      5063 1
3809      5064 1       Here on $GET errors, check for Record stream active error (RMSS_RSA)
3810      5065 1       If this error, WAIT until not active and try $GET again.
3811      5066 1       This recovers from AST I/O which can occur out of the middle
3812      5067 1       of synchronous I/O at non-AST level.
3813      5068 1
3814      5069 1   CALLING SEQUENCE:
3815      5070 1
3816      5071 1       JSB GET_ERROR ()
3817      5072 1
3818      5073 1   FORMAL PARAMETERS:
3819      5074 1
3820      5075 1       NONE
3821      5076 1
3822      5077 1   IMPLICIT INPUTS:
3823      5078 1
3824      5079 1       CCB                               Adr. of current LUB/ISB/RAB
3825      5080 1
3826      5081 1   IMPLICIT OUTPUTS:
3827      5082 1
3828      5083 1   ROUTINE VALUE:
3829      5084 1
3830      5085 1       NONE
3831      5086 1
3832      5087 1   SIDE EFFECTS:
3833      5088 1
3834      5089 1       If this is an INPUT LINE and a ^Z error, then just return and it will
3835      5090 1       be handled above.
3836      5091 1       SWAITs and then tries $GET again, until success or any error
3837      5092 1       except record stream active.
3838      5093 1
3839      5094 1
3840      5095 2   BEGIN
3841      5096 2
3842      5097 2   EXTERNAL REGISTER
3843      5098 2       CCB : REF BLOCK [. BYTE];
3844      5099 2
3845      5100 2
3846      5101 2       + Set the prompt buffer length to zero so that error followed by RESUME will not
3847      5102 2       keep concatenating the prompt buffer.
3848      5103 2
3849      5104 2       - CCB [RAB$B_PSZ] = 0;
3850      5105 2
3851      5106 2       + Is this INPUT LINE and only a ^Z in the record?
3852      5107 2
3853      5108 2
3854      5109 2       IF .CCB [ISBSB_STM_TYPE] EQL ISBK_ST_TY_INL AND .CCB [RAB$W_RSZ] EQLU 1 AND .(.CCB [RAB$L_RBF])<0, 8>
3855      5110 2       EQLU BASBK_CONTROL_Z
3856      5111 2
3857      5112 2       THEN
3858      5113 2       RETURN;
```

```

: 3859      5114  2 WHILE .CCB [RAB$L_STS] EQL RMSS_RSA DO
: 3860      5115  3   BEGIN
: 3861      5116  3     SWAIT (RAB = .CCB);
: 3862      5117  4     SGET (RAB = .CCB)
: 3863      5118  2   END;
: 3864      5119  2
: 3865      5120  2 IF NOT .CCB [RAB$L_STS]
: 3866      5121  2 THEN
: 3867      5122  2
: 3868      5123  2 !+ Check the input parameter to see if we should signal or stop.
: 3869      5124  2 !-
: 3870      5125  2
: 3871      5126  2
: 3872      5127  2   IF .SIGNAL_OR_STOP EQL K_SIGNAL
: 3873      5128  2     THEN BAS$SIGNAL_IO (BASSK_IOERR_REC)
: 3874      5129  2     ELSE BAS$STOP_IO (BASSK_IOERR_REC);
: 3875      5130  2
: 3876      5131  2
: 3877      5132  2
: 3878      5133  2 RETURN;
: 3879      5134  1 END;                                ! End of GET_ERROR

```

| 0000 00000 GET_ERROR: | | | | | |
|-----------------------|------|----|---------------------|-------|--------------------|
| | | | | .WORD | Save nothing |
| 20 | FF71 | 34 | AB 94 00002 | CLRB | 52(CC8) |
| | | | CB 91 00005 | CMPB | -143(CC8), #32 |
| 01 | 22 | 22 | OC 12 0000A | BNEQ | 1\$ |
| | | | AB B1 0000C | CMPW | 34(CC8), #1 |
| 1A | 28 | 28 | 06 12 00010 | BNEQ | 1\$ |
| | | | BB 91 00012 | CMPB | 040(CC8), #26 |
| 000182DA | 8F | 08 | 3D 13 00016 | BEQL | 4\$ |
| | | | D1 00018 1\$: 14 | CMPL | 8(CC8), #99034 |
| | | | 12 00020 | BNEQ | 2\$ |
| | | | 5B DD 00022 | PUSHL | CC8 |
| 00000000G | 00 | | 01 FB 00024 | CALLS | #1, SY\$SWAIT |
| 00000000G | 00 | | 5B DD 0002B | PUSHL | CC8 |
| | | | 01 FB 0002D | CALLS | #1, SY\$GET |
| | | | E2 11 00034 | BRB | 1\$ |
| 01 | 04 | 08 | AB E8 00036 2\$: 01 | BLBS | 8(CC8), 4\$ |
| | | | AC D1 0003A | CMPL | SIGNAL_OR_STOP, #1 |
| | | | 0B 12 0003E | BNEQ | 3\$ |
| 00000000G | 00 | 7E | 01 CE 00040 | MNEGL | #1, -(SP) |
| | | | 01 FB 00043 | CALLS | #1, BAS\$SIGNAL_IO |
| 00000000G | 00 | 7E | 04 0004A | RET | |
| | | | 01 CE 0004B 3\$: 01 | MNEGL | #1, -(SP) |
| | | | FB 0004E | CALLS | #1, BAS\$STOP_IO |
| | | | 04 00055 4\$: RET | RET | |

: Routine Size: 86 bytes. Routine Base: _BAS\$CODE + 0CF8

: 3880 5135 1 END
: 3881 5136 1

BASS\$REC_PROC
1-095

: 3882 5137 0 ELUDOM

K 12
16-Sep-1984 01:01:12
14-Sep-1984 11:56:35

VAX-11 Bliss-32 v4.0-742
[BASRTL.SRC]BASRECPRO.B32;1

Page 109
(42)

BASS\$REC_WF9==
BASS\$REC_WF1==
BASS\$REC_WF0==
BASS\$REC_WSL9
BASS\$REC_WSL1
BASS\$REC_WSL0

PSECT SUMMARY

| Name | Bytes | Attributes |
|-------------|--------------------------------------------------------------|------------|
| -BASS\$DATA | 6 NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON, PIC,ALIGN(2) | |
| -BASS\$CODE | 3406 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2) | |

Library Statistics

| File | Total | Symbols Loaded | Percent | Pages Mapped | Processing Time |
|-------------------------------------|-------|-------------------|---------|-----------------|--------------------|
| \$_255\$DUA28:[SYSLIB]STARLET.L32;1 | 9776 | 44 | 0 | 581 | 00:01.2 |

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:BASRECPRO/OBJ=OBJ\$:BASRECPRO MSRC\$:BASRECPRO/UPDATE=(ENH\$:BASRECPRO)

: Size: 3397 code + 15 data bytes
: Run Time: 01:16.8
: Elapsed Time: 02:50.9
: Lines/CPU Min: 4014
: Lexemes/CPU-Min: 26026
: Memory Used: 249 pages
: Compilation Complete

0030 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

